ICS NEWS
INFORMS Computing Society Newsletter

Inside This Issue...

"The Society is a great opportunity to get involved, meet some new people and generally combine your professional interests with a little fun."

Message from the Chair

Simge Küçükyavuz
Industrial Engineering and Management Sciences
Northwestern University, simge@northwestern.edu

What a year it has been! When I wrote my last "message from the chair" for our newsletter in January 2020, plans were underway for the ICS Conference in Tampa in January 2021. In just a couple of months, the world turned upside down! A once-in-a-century pandemic and the ongoing uncertainty forced us to make the difficult decision to postpone our biennial conference to January 23-25, 2022. Please mark your calendars and make this conference a high priority! We miss the in-person experience of conferences after one-too-many Zoom meetings, and look forward to seeing you in Tampa. This pandemic has also highlighted the importance of decision-making under uncertainty, with problems such as ventilator allocation, provider staffing, and vaccine distribution becoming ever more crucial. To this end, our members have been developing computational models and methods to solve such challenging problems.

In 2020, we launched a working group on quantum computing, chaired by Giacomo Nannicini, which will help provide ICS members with information and resources on this novel computing paradigm. We presented the inaugural Harvey J Greenberg Research Award during our virtual ICS Business Meeting, along with the traditional ICS Prize and ICS Student Paper Award. The fORged by Machines contest, led by Suvrajeet Sen, had a second successful year with the active engagement of our student members. So, we kept up the momentum in our work despite these trying times, thanks to all our members and officers. I hope you enjoy reading about some of the award-winning research in this newsletter.

Message from the Newsletter Editor

Yongjia Song
Department of Industrial Engineering
Clemson University, yongjis@clemson.edu

It is the time to share the news for the society again and it is my pleasure to put things together. In this newsletter, please be aware of the updates of the society officers, board of directors, *INFORMS Journal on Computing*, as well as research highlights and insights from the 2020 ICS awards. Special thanks to all who contributed to this newsletter! I will also greatly appreciate any comment or suggestion that you may have for the newsletter.

The Inaugural Harvey J. Greenberg Research Award

The award honors research excellence in the field of computation and operations research applications, especially those in emerging application fields. Honored research would focus on contributions that exhibit the promise of making a significant impact in the scope of OR/MS/Analytics practice.

Winners of the inaugural Harvey J. Greenberg Research Award: Danial Davarnia and Willem-Jan van Hoeve for their paper "Outer Approximation for Integer Nonlinear Programs via Decision Diagrams," forthcoming in Mathematical Programming Series A.

Committee members: Dorit Hochbaum (Chair), Pascal van Hentenryck, and Karla Hoffman.

The 2020 ICS Prize

The ICS Prize is an annual award for the best English language paper or group of related papers dealing with the Operations Research/Computer Science interface. **The 2020 prize was awarded to Samuel Burer and Renato D. C. Monteiro** for their pioneering work on low-rank semidefinite programming, as detailed in the papers:

(1) A Nonlinear Programming Algorithm for Solving Semidefinite Programs via Low-Rank Factorization, Mathematical Programming Series B 95: 329 – 357 (2003);
(2) Local Minima and Convergence in Low-Rank Semidefinite Programming, Mathematical Programming Series A 103, 427 – 444 (2005).

Committee members: Suvrajeet Sen (Chair), Fatma Kılınç-Karzan, Necdet Serhat Aybat

The 2020 ICS Best Student Paper Award

The ICS Student Paper Award is an annual award for the best paper at the interface of computing and operations research by a student author. **The 2020 winner was awarded to Tyler Perini, Georgia Institute of Technology.** Award-winning paper: "A Criterion Space Method for Biobjective Mixed Integer Programming: the Boxed Line Method".

Runner up: Margarita Castro (University of Toronto), "A Combinatorial Cut-and-Lift Procedure with an Application to 0-1 Chance Constraints". **Honorable mentions: Hussein Hazimeh, MIT**, "Sparse Regression at Scale: Branch-and-Bound rooted in First-Order Optimization", and **Prateek Srivastava, UT-Austin**, "A Robust Spectral Clustering Algorithm for Sub-Gaussian Mixture Models with Outliers".

Committee members: Claudia d'Ambrosio (Chair), Georgina Hall and Ruiwei Jiang

Message from the Editor-in-Chief of INFORMS Journal on Computing

by Alice Smith, Department of Industrial and Systems Engineering, Auburn University.
smithae@auburn.edu

Happy 2021 all INFORMS Computing Society members! The editorial board of the journal met virtually in November 2020 as part of the INFORMS Annual Meeting. Updates and statistics about the journal were presented and discussed, and I offer some as points of interest below:

- IJOC was first published in 1989 and some 1500+ articles have been published since then.
- There are ten technical areas.
- There are 67 associate editors along with the 10 area editors come from 15 countries over six continents.
- Over 30,000 downloads of IJOC papers occurred during the first three quarters of 2020.
- The desk reject rate was about 33% and the median time to first decision for papers undergoing review was about 130 days.
- Two areas have expanded. Stochastic Models now includes Reinforcement Learning and Software Tools now accepts full-length papers and has developed a full featured GitHub IJOC Repository for code and data.
- There is a 55% increase for 2021 in page budget for the journal to help with the publishing backlog.

So, as you can see, IJOC is healthy and progressive and this is thanks to the many editors, the INFORMS staff, the readers and authors of IJOC, and you, the supporting society of the journal.

IJOC is on Github!

by Ted Ralphs, Software Tools Area Editor at IJOC, Department of Industrial and Systems Engineering, Lehigh University. ted@lehigh.edu

Back in November 2020, Alice Smith, the Editor-in-Chief of our INFORMS Journal on Computing (IJOC) announced some enhancements to the publication process of the Software Tools area of IJOC, which will be rolling out to other areas over time. This short note is to follow up on that announcement in case you missed it. The most important aspect of the changes is that the INFORMS Journal on Computing is now on Github!

https://INFORMSJoc.github.io

The idea behind the hosting of digital artifacts, such as software and data, on Github is that such artifacts, when associated with accepted papers, are important archival research products in their own right. Thus, we are aiming to improve their visibility and availability. The Github organization provides a permanent home for such artifacts and offers a single, comprehensive site for readers who want to access them. An accompanying goal is to increase the impact of publication for authors by providing them with a separate DOI that allows these contributions to be recognized and cited as separate research works. Please explore the site and let us know what you think!

The editorial statement for the area has recently been updated to invite both full-length papers and shorter, focused papers. Our aim is to make IJOC the first choice for publishing software- and data-centric papers in the OR realm. Please join us in building what we expect will become the premier place to publish about software within the INFORMS community! More information can be found at https://pubsonline.informs.org/page/ijoc/editorial-statement

About fORged-by-Machines (2019 – 2020)

by Suvrajeet Sen, Data-Driven-Decisions Lab, University of Southern California.
sen@datadrivendecisions.org

The fORged-by-Machines Contest is a new effort which has been sponsored by Amazon Web Service (AWS) since 2019. This is intended as an effort to promote large scale computations for problems arising from the integration of predictive and prescriptive analytics. The role of the machine in this integration is undeniable, be it, for data from remote sensing and Internet of Things (IoT), or from decisions involving models whose exact representations may be so large as to compete with the number of atoms in the universe. As companies like AWS and others know all too well, talent to solve models of such magnitude require both clever OR/MS, as well as computing savvy. With Machine Learning becoming a household term, it seems like the appropriate time for INFORMS Computing Society to take the lead in promoting a contest which fORges the strengths of ORMS and Computing through their common interface, the machine. The total prize money for this contest is $1750, with $1000 for the Winner, $500 for the Runner-Up, and $250 for Honorable Mention. AWS also supported the costs associated with a meeting room at the conference hotel (in 2019), and to also host a small reception during the presentations. For obvious reasons, these aspects of the presentation session were not part of the activities in 2020.

This contest is open to teams of Ph.D. students anywhere in the world, provided one of their advisors is an INFORMS member, and can attest to the status of team members as Ph.D. students in their university. Other than being Ph.D. students, there are no other restrictions which have been announced for this contest. We have had entries from a variety of countries, including China, Germany, Korea,

Turkey, U.K., and of course the U.S. The prize winners for each of the years is given below.

- 2019: Winner: Priyadarshan Patil (Univ. of Texas, Austin); Runner-up: Saeed Chavoshi and Rahman Khorramfar (NC State Univ.); Honorable Mention: Roshanak Khalegi and Juan Xu (Univ. of Illinois)
- 2020: Winner: Santiago Nieto-Isaza, Emanuel Herrmann (Tech.Univ.- Munich); Runner-up: Saranthorn Phusingha, Alexandra Blennerhasset (Univ. of Edinburgh); Honorable Mention: Barbara Rodrigues and Malte Billen (Univ. of Edinburgh)

In these first couple of years, we have had a team of three judges who have down-selected submissions to three teams, and these three are designated as "finalists". The finalists are invited to make presentations to the judges. All finalists are judged as winners, although their rank among the finalists is decided after a presentation and Q&A with the judges. The first of these contests was held at the INFORMS Annual Conference in Seattle (2019), and the next one was held virtually at the INFORMS Annual Conference (Online) in 2020. It goes without saying that the contest itself would not have been possible without the support of INFORMS members affiliated with AWS. In particular, Kerem Bulbul (AWS) and Semih Atakan (Amazon) were instrumental in launching this effort. We are very grateful to the judges (John Carlsson (USC), Anton Kleywegt (Ga. Tech.), and Mauricio Resende (Amazon)) for their clarity of vision, and dedication to the theme of this contest. As always, the INFORMS staff has been a pleasure to work with; their professionalism and commitment to a high-quality experience (including the Zoom era) remains unparalleled. Finally, no report on this contest would be complete without mentioning the support and commitment of the ICS chair, Simge Küçükyavuz. Her steadfast vision is fundamental to growth of both computing and ICS within INFORMS. Finally, I must take this opportunity to thank my current and former Ph.D. students who keep pushing the boundaries of OR/MS computing, and keep me thoroughly engaged in the transformation of Stochastic Optimization from an essentially mathematical pursuit to its computational realization, and making this sub-area vibrant for "prime-time" applications.

COIN-OR: A 20-year Retrospective

by Brady Hunsaker bhunsaker@google.com
Lou Hafer lou@sfu.ca
Ted Ralphs ted@lehigh.edu
Matt Saltzman mjs@clemson.edu [1]

Build it, and they will come. That's what they say. In the early aughts, a hardy band of IBMers and a few intrepid outsiders put this to the test. Did it work? How did that cute and promising infant grow to be the truculent young adult it is today? Twenty years on, we look back to provide a few insights.

1 A (Very) Brief History

1.1 In the beginning?

The seed for what became the COIN-OR initiative was planted and germinated by some observations about the state and trajectory of research in computational optimization in the late 90s. Increasingly, complex algorithms were being built out of simpler parts. (Think of solvers for mixed integer linear optimization problems (MILPs) constructed from solvers for linear optimization problems (LPs).) Without open-source software, many wheels were being reinvented, stifling progress. At the same time, there was no real venue for "publication" of software. Establishing such a forum would kill two birds with one stone. The pithy, elevator version of the mission of what was soon to be created became "to be for software what the OR literature is for theory."

And so it was that in August of 2000, IBM announced what was then billed as "The COIN-OR Initiative" at ISMP. Back then, COIN-OR stood for *Common Optimization INterface for Operations Research*, and the focus was to be on developing and distributing open-source software for solving optimization problems. To show its commitment to the concept and to get things rolling, IBM seeded the initial repository with some projects it had developed with a few external partners and hosted the project's infrastructure on its developerWorks platform. The first outside contribution, Open Tabu Search (OTS), came in quickly at the end of October 2000.

Initially, COIN-OR had more "parts" than "wholes," which was kind of the idea. Grab a few cut generators, a branch and bound framework, and an interface to existing LP solvers, and you could build yourself a MILP solver! The initiative enjoyed strong support from IBM in the early years. The IBMers quickly ramped up their contributions, with the first official "release" of the existing suite of tools coming one year later in August 2001. In April of 2002, the venerable BDFL (Benevolent Dictator For Life: a common nickname for the spiritual leader of an open-source project) of COIN-OR, John Forrest, posted a message to the mailing list with the subject "Anyone want an Open Source simplex solver?" There, he detailed, in his usual humble and understated way, how he had set out to test the Gomory cut generator in CGL and at a particular stage "realized that if [he] did not stop soon [he] would have a

[1] The authors are the founding board members of the COIN-OR Foundation, Inc., which was created in 2004 as steward of the COIN-OR initiative. The "we" throughout this article refers to a much larger cohort of volunteers and supporters who have been involved in the project over time.

simplex solver!". Thus, the Clp project was born and was followed a month later by the addition of Ipopt. January 2003 saw the introduction of Sbb (Simple Branch and Bound), the first draft of what later grew into Cbc.

At around this time, plans were being made to fulfill one of IBM's explicitly stated goals for the project – to move COIN-OR off IBM's servers and develop it into a community-based project. In November 2002, the INFORMS Board approved a proposal to host COIN-OR for three years. The fledgling was perched on the edge of the corporate nest. But could it fly? No one knew for sure.

1.2 COIN-OR Foundation, Inc.

Discussions began in earnest about what a community-based COIN-OR would look like. How should it be governed? And by whom? Should there be a formal submission and review process for inclusion in the repository? Should the nascent organization try to raise money to support activities? Should it be an official non-profit entity? What about licensing and ownership of intellectual property? We got a crash course, and dozens of meetings later, with copious use of IBM's teleconference facility (no Zoom in 2002!), we had answers to some questions, but many remained.

One thing quickly became certain, as INFORMS insisted as a condition for hosting that we form a non-profit entity for liability reasons. On March 1, 2004, we became the COIN-OR Foundation, Inc., a Maryland corporation. Along the way, we rebranded with a more inclusive interpretation of the by-then well-established acronym: **COmputational INfrastructure for Operations Research** (see what we did there?). The inaugural business meeting of the Foundation and one of the early tutorial workshops on how to get started with COIN-OR took place at the joint CORS/INFORMS Conference in May 2004. It took another year to acquire 501(c)(3) public charity status; our application was filed with the IRS in March 2005 and approved in November 2005.

Things began to move quickly in the ensuing years. Clp released v1.0 in October 2004 and Sbb rebranded to Cbc a few weeks later. The INFORMS Annual Meeting in November marked the first appearance of the COIN-OR booth in the trade show. INFORMS 2005 saw the first COIN-OR Cup award (winner: John Forrest), the announcement of Cbc v1.0, and, most importantly, the debut of the COIN-OR chocolate coins!

2006 saw a number of technical advances. The build harness that has grown into today's BuildTools configuration scaffolding debuted in May. We moved from CVS to Subversion and TRAC for project management. Bonmin, a nonlinear mixed-integer optimization code built using COIN-OR components, appeared in the repository in July. 2006 also brought the COIN-OR logo contest that gave us our current logo, and it marked the formation of the long-standing strategic partnership between the foundation and the INFORMS Computing Society.

1.3 Fast forward to the present

We eventually hit our stride, notwithstanding the ever-present feeling that COIN-OR hadn't quite fully arrived. While the Strategic Leadership Board (SLB) debated seemingly mundane questions of strategy and growth behind the scenes, the Technical Leadership Council (TLC) managed the front-facing infrastructure and the review process. We continued to attract quality contributions, and every so often, a volunteer with the enthusiasm and willingness to join one of our boards would wander in. To those unsung volunteers who have contributed to COIN-OR over the years, we owe a huge debt of gratitude. Thank you!

Over the intervening years, the repository grew to 70 projects, and software was distributed to thousands of users from our servers. COIN-OR hosted 15 Cup ceremonies, convened 15 meetings of its membership, wrote 15 annual reports, put on countless tutorials and workshops, and hosted two multi-day workshops. The most recent of these was a week-long workshop at the Institute for Mathematics and its Applications at the University of Minnesota. This was followed by several "coding sprints" focused around some of the more recent efforts associated with COIN-OR, such as the development of the new linear optimization framework HiGHs and the JSON-based data interchange format MOSDEX.

In 2014, the entire team was deeply honored when INFORMS awarded the Impact Prize to COIN-OR. Although nine names appear on the certificate, we all felt that credit was due to many more people and organizations than that. It is satisfying that these days, the answer to the question "Have you heard about COIN-OR?" is more likely to be a quizzical "of course, who hasn't?" rather than the blank expression of yesteryear. But have they really come? Has it been worth the enormous effort? What have the impacts been? Has the Foundation succeeded in achieving its goals? Well, yes? and no. In the remainder of this article, we provide some perspective on these and other questions.

2 Reflections

2.1 How hard can it be?

It seemed like a good idea at the time, but it turns out that operating a non-profit foundation is actually a lot of work! And it should probably be obvious that researchers (be they from academia or industry) are not the right ones to put in charge of business operations. Nevertheless, when we designed the COIN-OR Foundation's structure, one of the things we got right was to separate the care for the institution, such as funding and legal requirements, from care for the software and infrastructure that is its reason to exist. The former is handled by the aforementioned SLB, while the latter is handled by the TLC. While each board occasionally became mired in the minutiae of its mandate, the decision to keep them separate meant they were

each able to move forward more or less independently on different tracks.

On the technical side, the goal was initially (and to a large extent remains) to provide the infrastructure to allow others to easily do the core work of implementing algorithms, insulating them from the effort of developing the infrastructure for building, testing, ensuring portability, maintaining issue trackers, making releases, archiving, making source code available, maintaining web infrastructure, *etc.* For the TLC, keeping up with technology has meant chasing a constantly moving target. Over the years, we've migrated through three project management systems and three version control systems, ported code to work with countless compilers and operating systems, moved the entire contents of the repository to new servers four times, utilized three different continuous integration systems, migrated Web content across four different servers and from plain HTML to a CMS, and the list goes on. From the beginning, it's been a constant battle to keep up with advances in technology while making the time to do the work we really love—implementing algorithms!

Meanwhile, on the strategic side, the SLB's effort has gone into questions of how to promote COIN-OR; how to implement the review process; how to increase participation; how to manage potential legal issues; how to raise funds; whether the Foundation itself should support development or continue with volunteers. Easy to say, but as any of you who have grappled with such matters will attest, the devil is in the details.

2.2 Did they come?

Uptake was definitely slow at first, but we always knew that we would need to remain focused on the long game. Working with the tools as they were in the beginning took a hacker's mentality and a developer's knowledge. There was no StackOverflow. There were no installers, no binaries. Development was geared towards Linux, which itself was still primarily the domain of technological adventurers. So while additional contributions filtered in and the repository grew, the foundation leadership, with its dual strategic and technical boards, slowly chipped away at the barriers to entry and set a course for the future.

Over the years, participation and awareness have increased dramatically. Still, we've struggled to engage the community directly and to identify exactly who is using COIN-OR codes and what the value proposition really is. The growth of COIN-OR has largely mirrored and benefitted from larger trends in software and technology. The recent move to Github has allowed users to participate in a more frictionless fashion, using tools they've become accustomed to. The advent of popular, extensible high-level languages, such as Python and Julia, means that there are now ways to interface to COIN-OR codes from almost every commonly-used development and modeling environment. In short, the barriers to entry are lower. On the flip side, this leaves us with many users we have no way of knowing about and (a larger problem from our point of view) many users who have no way of knowing about *us*.

One significant "they" which has *not* come are developers committed to helping ensure the long-term health of the more utilitarian parts of the COIN-OR code base. We have been struggling for many years with an exodus of core developers as they move on to new challenges and phases of their lives. Finding ways to lower the barriers to entry and increase the rewards for new developers to participate is something we have yet to do effectively.

2.3 What have we achieved?

Looking back, the report card is mixed. We have succeeded on some fronts and not on others. Above all, COIN-OR provided, and continues to provide, a focus on and infrastructure for nurturing the development of quality open-source software for operations research. The worth of that contribution is demonstrated in several ways. Most obviously, the existence of the open-source components led directly to the rapid development of increasingly sophisticated tools for both optimization and modeling, each building on the base provided by earlier contributions. Less obviously, the use of COIN-OR software in other open-source projects, such as OpenOffice, has proliferated. Even less obviously, the COIN-OR repository provides a library for the art (dare we call it that?) of programming that future developers can read and learn from.

We have largely succeeded (in concert with many related efforts and trends) in raising the profile of computational research and increasing the academic credit associated with software development activities. Although there is still a long way to go, it looks far easier to get academic credit for software these days than 20 years ago. This progress is reflected in the emergence of new journals, such as Mathematical Programming Computation, and the Software Tools area of the INFORMS Journal on Computing. Naturally, there were many forces simultaneously at work that promoted this slow change, but we hope COIN-OR had at least some role to play. At the very least, the existence of the codes in COIN-OR has enabled a substantial amount of computational research that would not have otherwise been possible.

On the negative side, we have so far failed to find significant institutional funding for the maintenance and development of COIN-OR. And we have so far largely failed to attract a critical mass of developers with the time and energy to maintain the existing code base. The answer to how to make the reward commensurate with the effort for new recruits still eludes us.

2.4 Whither from here?

As we complete our latest migration to Github, WordPress, and GSuite for Nonprofits, the effort involved in maintenance of infrastructure is lessening and we are contemplating a return to our original vision: to become a

well-respected, peer-reviewed publication venue for open-source software related to operations research. The question is how to do this while also maintaining our identity as a resource for—and a community of—both developers and users within the operations research community. What role do the Foundation and the people involved in it have in continuing to maintain and develop the core set of tools that have been built over the last 20 years, as their original authors move on to other things?

And so, we are looking to the community for guidance. What do you see as the value proposition of COIN-OR? How has it impacted your work? What has it allowed you to do? Is it worth our collective efforts to support? If so, we challenge you to find a way of giving back. First and foremost, help us raise awareness. Cite the COIN-OR codes you find useful (in addition to the papers describing them) and give us a shout-out on Twitter @coin_or! Make others aware that they may be using COIN-OR without knowing it. Did you know Mathematica embeds Clp and that the solver in OpenOffice is powered by COIN-OR? Volunteer some of your time as a COIN-OR board member or to help with technical tasks. Best of all, polish up that proof-of-concept code that backs up your publication and make it available to other researchers. After all, the engine of scientific progress is reproducibility and the engine of reproducibility is open source!

Rest assured, we are not going anywhere soon. In fact, a copy of the entire COIN-OR repository has been deposited for posterity somewhere beneath the arctic tundra (for real![2]). But if you made it this far, thanks for reading and we welcome your input on how to move forward and serve the community's needs. It's a big tent and all are welcome. We'll leave the light on for ya!

Research Highlight: Low-Rank Methods for Semidefinite Programming

by SAMUEL BURER, DEPARTMENT OF BUSINESS ANALYTICS, UNIVERSITY OF IOWA, SAMUEL-BURER@UIOWA.EDU AND RENATO D.C. MONTEIRO, DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING, GEORGIA TECH, RENATO.MONTEIRO@ISYE.GATECH.EDU

In [4], published in 2003, the authors introduced the low-rank idea for solving semidefinite programs (SDPs), which replaces the primal matrix variable $X \succeq 0$ with a low-rank rectangular factorization RR^T, resulting in a specially structured, nonconvex quadratically constrained quadratic program (QCQP) in the variable R:

$$\begin{aligned} &\min\{C \bullet X : A_i \bullet X = b_i \; \forall \, i, X \succeq 0\} \\ \rightarrow \quad &\min\{C \bullet RR^T : A_i \bullet RR^T = b_i \; \forall \, i\}. \end{aligned}$$

Although perhaps this change of variables is non-intuitive, it has the potential to alleviate the computational drawbacks of interior-point methods for SDP—in terms of time, memory, and the ability to exploit sparsity and structure in the problem data.

An important point is that the number of columns in R should be taken large enough so that not all optimal solutions of the SDP are eliminated by the change of variables. For this, the authors drew on the theory of the geometry of spectahedra (i.e., SDP feasible sets) [9, 1] to choose the number of columns minimally, i.e., large enough but not too large. The authors then derived the standard first- and second-order necessary optimality conditions and highlighted the interesting geometrical fact that the QCQP can have no strict local minima. Nevertheless, the authors established new checkable, sufficient second-order conditions guaranteeing global optimality of the QCQP—and hence of the primal SDP—and showed how these conditions closely relate to dual optimality of the SDP.

The authors finally developed a first-order augmented Lagrangian algorithm with inexpensive function and gradient evaluations, which uses the low-rank idea to solve large-scale instances of the SDP relaxations of several combinatorial optimization problems, while simultaneously taking into account sparsity and structure in the problem data. In particular, the authors showed that the implementation outperformed both interior-point methods and other first-order methods on these problem classes, sometimes by several orders of magnitude.

However, even though the algorithm performed well in practice, as may be clear from the above description, our low-rank algorithm was properly seen as a heuristic for solving SDPs because there was no formal theory guaranteeing that the algorithm would converge to a global optimal solution of the QCQP/SDP. This led to a lingering research question: why should solving the nonconvex QCQP in place of the convex SDP work well in practice; shouldn't it get trapped in non-global local minima?

In [5], the authors provided a partial theoretical justification for why the low-rank approach converges to global optimal solutions in practice. A key innovation in the paper is a careful comparison of the rank-constrained primal, i.e., $X \succeq 0$ with $rank(X) \leq r$, and the QCQP over R with r columns. First, realizing that the $rank(X) \leq r$ constraint can indeed introduce local minima in the primal, the change of variables $X = RR^T$ nevertheless does not introduce *additional* local minima in the QCQP. In a sense, the change of variables doesn't make the problem any worse. Second,

[2] https://archiveprogram.github.com/arctic-vault/

if X is a local minimum of the rank-constrained primal, then one of two things must occur: either X is an optimal extreme point of the original SDP (the "good case"), or X lies within a face of the feasible set of the SDP, which is constant, or flat, with respect to the linear objective function (the "flat case"). Note that the flat case cannot occur unless there does in fact exist a positive-dimension face having constant objective value, and even if that does occur, it is unstable under perturbations of the objective function. This by itself provides some intuition as to why the low-rank algorithm works in practice, i.e., because the good case is much more likely to occur for "random" objectives.

In addition to establishing this intuition, the authors further analyzed a slight variant of the augmented Lagrangian algorithm from [4], having the following convergence guarantee: assuming that a local minimum is obtained at each stage of the augmented Lagrangian algorithm, every accumulation point $X = RR^T$ of the resulting sequence is globally optimal. Note that the analysis of this algorithm was still incomplete in two respects: (i) it did not directly deal with the flat case, but rather cleverly avoided it; (ii) it assumed that a local minimum was obtained at each stage of the augmented Lagrangian algorithm. More on these deficiencies below.

The authors also demonstrated additional, strong computational results—especially on the largest SDP relaxations of the quadratic assignment problem solved up to that point in time—and discussed how to obtain valid dual bounds from the primal-only algorithm.

Since the publication of [4, 5], the low-rank idea has become a standard approach for the design of algorithms to solve large-scale, structured SDPs, and it has also been extended to other problem classes, especially problems where a non-symmetric factorization $Y = RS^T$ is appropriate [8]. A quick examination of papers citing [4, 5] reveals applications in rank-minimization, network clustering, image analysis, quantum chemistry, robust PCA, etc. In particular, the low-rank idea has enjoyed strong interest since about 2015, apparently for two reasons.

First, the low-rank idea has found a home in the machine-learning community, where it fits well within the current trend of examining strong guarantees for nonconvex optimization problems. In addition to low-rank approaches for SDP, there are a number of other cases in machine learning for which one can reliably compute global optimal solutions, even though the optimization problem is highly nonconvex; see [11] for example. As a result, there is a growing sense in machine learning that nonconvexity is not necessarily something to fear. Training neural networks is another example, although admittedly the nonconvexity inherent in neural networks is quite different than low-rank nonconvexity. So [4, 5] are a fundamental contribution to this important research trend in machine learning.

Second, the theory of why the low-rank algorithm works well for SDP has been largely settled in recent years. In particular, the two deficiencies (i) and (ii) mentioned above have been addressed in [3], where the authors show that, when the QCQP feasible set of the primal SDP is a smooth manifold, then for almost all cost functions, a second-order stationary point R of the QCQP is globally optimal, i.e., the necessary second-order conditions are sufficient for global optimality. Moreover, manifold-based algorithms deliver second-order stationary points. These strong guarantees apply in many applications. For example, they apply for a particular SDP at the heart of an application in robotics called "simultaneous localization and mapping" (SLAM), where the low-rank approach is one of the most popular ones in practice [10]. Additional papers have extended results of this type, e.g., [6, 12, 7, 2]. So the theoretical advancements in the low-rank area continue to evolve in exciting ways.

References

[1] A. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete Computational Geometry*, 13:189–202, 1995.

[2] S. Bhojanapalli, B. Neyshabur, and N. Srebro. Global optimality of local search for low rank matrix recovery. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 3873–3881. Curran Associates, Inc., 2016.

[3] N. Boumal, V. Voroninski, and A. S. Bandeira. Deterministic guarantees for Burer-Monteiro factorizations of smooth semidefinite programs. *Comm. Pure Appl. Math.*, 73(3):581–608, 2020.

[4] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (Series B)*, 95:329–357, 2003.

[5] S. Burer and R. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[6] D. Cifuentes and A. Moitra. Polynomial time guarantees for the burer-monteiro method, 2019.

[7] W. Ha, H. Liu, and R. F. Barber. An equivalence between critical points for rank constraints versus low-rank factorizations. *SIAM J. Optim.*, 30(4):2927–2955, 2020.

[8] D. Park, A. Kyrillidis, C. Carmanis, and S. Sanghavi. Non-square matrix sensing without spurious local minima

via the Burer-Monteiro approach. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 65–74, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[9] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Math. Oper. Res.*, 23:339–358, 1998.

[10] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[11] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via procrustes flow. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 964–973, New York, New York, USA, 20–22 Jun 2016. PMLR.

[12] I. Waldspurger and A. Waters. Rank optimality for the Burer-Monteiro factorization. *SIAM J. Optim.*, 30(3):2577–2602, 2020.

Research Highlight: Outer Approximation for Integer Nonlinear Programs via Decision Diagrams

by DANIAL DAVARNIA, DEPARTMENT OF INDUSTRIAL AND MANUFACTURING SYSTEMS ENGINEERING, IOWA STATE UNIVERSITY, DAVARNIA@IASTATE.EDU AND WILLEM-JAN VAN HOEVE, TEPPER SCHOOL OF BUSINESS, CARNEGIE MELLON UNIVERSITY, VANHOEVE@ANDREW.CMU.EDU

We would like to thank the INFORMS Computing Society Harvey J. Greenberg research award committee members, Dorit Hochbaum, Pascal van Hentenryck, and Karla Hoffman, for the honor of being selected as the winner of this award. We would also like to thank the ICS newsletter editor, Yongjia Song, for the kind invitation to share this research highlight with the ICS community. This summary is based on our work [6].

3 Introduction

As a new technique to solve integer programs (IPs), decision diagrams (DDs) provide a graphical model that compactly represents the solution set together with the objective rates of the IP; see [2] for an introduction. In principle, DDs share similar recursive modeling as that used in dynamic programming (DP). However, instead of relying on the complete enumeration of the entire state space, DDs employ key notions of IP such as relaxation, restriction and branching search to overcome the drastic dimensionality growth of DP. Applications of DDs in various areas show the effectiveness of using DDs in optimization; see [4, 5, 8, 9, 10]. Currently, designing a DD package that contains effective relaxation, restriction and branching search relies on the existence of special structural properties in the problem that allow for recursive formulations; see for instance the approach used in [1] for set covering and stable set problems. This limitation encourages development of alternative DD techniques that can be applied to a broader class of problems with more general structure.

In this paper, we undertake this task by introducing a novel outer-approximation (OA) framework [3], composed of a DD constructor and a subgradient-type cut-generator, for a general class of IPs. The significance of this work is as follows. (i) The subgradient vectors in the cut-generator are derived through computation of the shortest-path on the graph of DD, making it a derivative-free approach, unlike the linearization cuts used in classical OA methods. (ii) The DD construction and the cut-generating methods can both be applied to constraints of any form including noncovnex, nonsmooth and even black-box models, providing a notable advantage over traditional OA techniques that require convexity to guarantee global validity of cuts. (iii) The framework applies to IPs with general structures, a property that mitigates the dimensionality issue of traditional DP models as well as the design limitation of typical DD models. (iv) Unlike the standard approach in factorable programming that decomposes complicated terms of a constraint separately, the DD representation considers the direct interaction of complicated terms in their base constraint, allowing for generation of stronger cuts. (v) The proposed cut-generator is remarkably faster than conventional cut-generating linear programs in finding a violated inequality during separation. Computational experiments are conducted on both synthetic and benchmark instances from different application areas. The bounds obtained from the developed framework exhibit significant gap improvements compared to those obtained from modern local and global

solvers.

4 Background on Decision Diagrams

Define $N := \{1, \dots, n\}$. We denote a DD by $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(.))$, where $\mathcal{U}$ represents the set of nodes and $\mathcal{A}$ represents the set of arcs in a top-down directed multi-graph, and function $l : \mathcal{A} \to \mathbb{Z}$ indicates the label of arcs in $\mathcal{A}$. The multi-graph induced by $\mathcal{D}$ is composed of n arc layers $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n$, and $n+1$ node layers $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_{n+1}$. Node layer $\mathcal{U}_1$ contains a single *source* node s, and the node layer $\mathcal{U}_{n+1}$ contains a single *terminal* node t. DD $\mathcal{D}$ represents a set of points of the form $\boldsymbol{x} = (x_1, \dots, x_n)$ with the following encoding. The label $l(a)$ of each arc $a \in \mathcal{A}_j$, for $j \in N$, represents the value of x_j. Each node in layer $\mathcal{U}_j$ has a maximum outdegree equal to the number of distinct integer values in the domain of variable x_j. This definition implies that each arc-specified path $\mathrm{P} = (a_1, \dots, a_n) \in \mathcal{A}_1 \times \dots \times \mathcal{A}_n$ from s to t encodes an integral assignment to vector $\boldsymbol{x} = (x_1, \dots, x_n)$ such that $x_j = l(a_j)$ for all $j \in N$. The collection of points encoded by all paths of $\mathcal{D}$ is referred to as the solution set of $\mathcal{D}$ denoted by $\mathrm{Sol}(\mathcal{D})$.

Using the above descriptions, we can model IPs with DDs. Consider the bounded IP $z^* = \max\{f(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{P}\}$, where $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{P} \subseteq \mathbb{Z}^n$. To model the above IP with a DD, we first form a *weighted* DD, denoted by $[\mathcal{D}|w(.)]$, where (i) $\mathcal{D}$ represents an encoding of solutions of $\mathcal{P}$, and (ii) $w(.) : \mathcal{A} \to \mathbb{R}$ is a weight function associated with arcs of $\mathcal{D}$ so that for each s-t path $\mathrm{P} = (a_1, \dots, a_n)$, its weight $w(\mathrm{P}) := \sum_{i=1}^{n} w(a_i)$ is equal to $f(\boldsymbol{x}^{\mathrm{P}})$, the objective value of the integral solution corresponding to P. The above definition implies that z^* is equal to the weight of the longest path from s to t in the corresponding weighted graph. Note that the longest path can be obtained in $\mathcal{O}(|\mathcal{U}| + |\mathcal{A}|)$, as it is solved over a directed acyclic graph. A comprehensive review of DDs can be found in [2].

5 Cut-Generating Methods

The first step to design an outer approximation framework based on DDs is developing a cut-generating oracle. In this section, we present two methods to derive valid inequalities for the convex hull of the feasible solutions of DDs. One is based on a cut-generating linear program (CGLP) and the other is based on a subgradient-type method. Consider a DD $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(.))$ with solution set $\mathrm{Sol}(\mathcal{D}) = \mathcal{P}$. Proposition 1 gives a CGLP to separate a given point $\bar{\boldsymbol{x}}$ from $\mathrm{conv}(\mathcal{P})$.

Proposition 1. *Consider a point $\bar{\boldsymbol{x}} \in \mathbb{R}^n$, and define*

$$\omega^* = \max \sum_{k \in N} \bar{x}_k \gamma_k - \theta_t \tag{1a}$$

$$\text{s.t. } \theta_{t(a)} - \theta_{h(a)} + l_a \gamma_k \leq 0, \qquad \forall k \in N, a \in \mathcal{A}_k \tag{1b}$$

$$\theta_s = 0 \tag{1c}$$

$$C(\boldsymbol{\theta}, \boldsymbol{\gamma}) \leq 0, \tag{1d}$$

where $t(a)$ and $h(a)$ represent the tail and the head node of arc a, and $C(\boldsymbol{\theta}, \boldsymbol{\gamma}) \leq 0$ is a typical normalization constraint. Then, $\bar{\boldsymbol{x}} \in \mathrm{conv}(\mathcal{P})$ if and only if $\omega^* = 0$. Otherwise, $\bar{\boldsymbol{x}}$ can be separated from $\mathrm{conv}(\mathcal{P})$ via $\sum_{k \in N} x_k \gamma_k^* \leq \theta_t^*$ where $(\boldsymbol{\theta}^*; \boldsymbol{\gamma}^*)$ is an optimal solution of (1). □

The shortcoming of the above CGLP is its computational burden for large size DDs that are common in practice. To mitigate this computational difficulty, we next derive a cut-generating method based on the projected subgradient algorithm for convex programs. This method has the advantage of exploiting the network structure of DDs to generate valid inequalities more effectively than the CGLP. The idea is to reformulate (1) as a bilevel program, which contains variables $\boldsymbol{\gamma}$ at the higher level and has two properties: (i) its objective function is piecewise-linear and concave; (ii) its subgradient at each point $\bar{\boldsymbol{\gamma}}$ can be obtained by calculating the longest path on an appropriately-weighted DD. Using these properties, we design Algorithm 1, which can be shown to converge to an optimal solution of (1).

The geometric intuition of Algorithm 1 is as follows. The key is to view $\boldsymbol{\gamma}^{(\tau)}$ as the normal vector of a valid inequality of the form $\boldsymbol{\gamma}^{(\tau)}\boldsymbol{x} \leq \psi$ whose right-hand-side ψ needs to be calculated. This calculation is done through lines 3 and 4 of the algorithm. These lines compute the minimum right-hand-side value for $\boldsymbol{\gamma}^{(\tau)}\boldsymbol{x} \leq \psi$ to be valid for $\mathrm{conv}(\mathrm{Sol}(\mathcal{D}))$, i.e., $\boldsymbol{\gamma}^{(\tau)}\boldsymbol{x} \leq \boldsymbol{\gamma}^{(\tau)}\boldsymbol{x}^{\mathrm{P}^{(\tau)}}$ is the strongest valid inequality with the fixed normal vector $\boldsymbol{\gamma}^{(\tau)}$. We refer to this inequality as *normal inequality* corresponding to $\boldsymbol{\gamma}^{(\tau)}$. It is easy to verify that the closure of normal inequalities for all vectors $\boldsymbol{\gamma}^{(\tau)}$ describes $\mathrm{conv}(\mathrm{Sol}(\mathcal{D}))$. With this perspective, the algorithm searches for the normal vector of a normal inequality that separates $\bar{\boldsymbol{x}}$. The updating step in this search is obtained by "tilting" the current normal vector $\boldsymbol{\gamma}^{(\tau)}$ toward the direction of the subgradient vector connecting $\boldsymbol{x}^{\mathrm{P}^{(\tau)}}$ (which can be interpreted as a feasible point that is tight at the current normal inequality) to $\bar{\boldsymbol{x}}$ (the point to be separated.)

Algorithm 1 A cut-generating algorithm based on a projected subgradient method

Input: A DD $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(.))$ and a point $\bar{\boldsymbol{x}}$

Initialize $\tau \leftarrow 0$, $\boldsymbol{\gamma}^{(0)} \in \mathbb{R}^n$, $\tau^* \leftarrow 0$, $\Delta^* \leftarrow 0$
while The stopping criterion is NOT met **do**
  Create a weighted DD $[\mathcal{D}|w(.)]$ with arc weights $w_a = l_a \gamma_k^{(\tau)}$ for all $k \in N$ and $a \in \mathcal{A}_k$
  Find a longest s-t path $\mathrm{P}^{(\tau)}$ in $[\mathcal{D}|w(.)]$ and compute its encoding point $\boldsymbol{x}^{\mathrm{P}^{(\tau)}}$
  if $\boldsymbol{\gamma}^{(\tau)}(\bar{\boldsymbol{x}} - \boldsymbol{x}^{\mathrm{P}^{(\tau)}}) > \max\{0, \Delta^*\}$ **then**
    Update $\tau^* \leftarrow \tau$ and $\Delta^* \leftarrow \boldsymbol{\gamma}^{(\tau)}(\bar{\boldsymbol{x}} - \boldsymbol{x}^{\mathrm{P}^{(\tau)}})$
  end if
  Update $\boldsymbol{\phi}^{(\tau+1)} \leftarrow \boldsymbol{\gamma}^{(\tau)} + \rho_\tau(\bar{\boldsymbol{x}} - \boldsymbol{x}^{\mathrm{P}^{(\tau)}})$ for step size ρ_τ
  Find the projection $\boldsymbol{\gamma}^{(\tau+1)}$ of $\boldsymbol{\phi}^{(\tau+1)}$ onto the set defined by $C(\boldsymbol{\gamma}) \leq 0$
  $\tau \leftarrow \tau + 1$
end while
if $\Delta^* > 0$ **then**
  Return inequality $\boldsymbol{\gamma}^{(\tau^*)}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{P}^{(\tau^*)}}) \leq 0$
end if

6 Outer Approximation

The cut-generating methods described in Section 5 can be used to form an *outer-approximation* for the following integer nonlinear program (INLP)

$$\begin{aligned} \max \quad & \boldsymbol{c}^\intercal \boldsymbol{x} \\ \text{s.t.} \quad & g^j(\boldsymbol{x}) \leq 0, && \forall j \in J \\ & \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}] \cap \mathbb{Z}^n, \end{aligned} \tag{2}$$

where $g^j(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is a general nonlinear function. We refer to this INLP as $(\mathcal{E})$.

It is well-known that when $g^j(\boldsymbol{x})$'s are convex and sufficiently smooth, a linear outer-approximation model can be constructed by replacing nonlinear constraints (2) with the so-called linearization cuts of the form $g^j(\bar{\boldsymbol{x}}) + \nabla g^j(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}}) \leq 0$ for all points $\bar{\boldsymbol{x}}$ that violate $g^j(\boldsymbol{x}) \leq 0$; see [3, 7] for a survey.

Inspired by this model, we design a novel outer-approximation framework that generalizes to problems with non-convex constraint functions. To this end, we define the set of discrete points described by constraint j of (2) over variable domains as $\mathcal{G}^j_{\mathbb{Z}} := \{\boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}] \cap \mathbb{Z}^n \,|\, g^j(\boldsymbol{x}) \leq 0\}$. Let $\mathcal{D}_j$ be the DD representing the solutions of $\mathcal{G}^j_{\mathbb{Z}}$, i.e., $\mathrm{Sol}(\mathcal{D}_j) = \mathcal{G}^j_{\mathbb{Z}}$. Next, we define $(\mathcal{E}^D(\mathcal{K}))$ for any finite set $\mathcal{K} \subseteq \mathbb{R}^n$ as

$$\begin{aligned} \max \quad & \boldsymbol{c}^\intercal \boldsymbol{x} \\ \text{s.t.} \quad & h_j(\bar{\boldsymbol{x}}, \boldsymbol{x}) \leq 0, && \forall \bar{\boldsymbol{x}} \in \mathcal{K}, \quad j \in \bar{J}(\bar{\boldsymbol{x}}) \\ & \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}] \cap \mathbb{Z}^n, \end{aligned} \tag{3}$$

where the inequality (3) is obtained via the CGLP of Proposition 1 to separate point $\bar{\boldsymbol{x}} \in \mathcal{K}$ from the convex hull of the solution set $\mathrm{Sol}(\mathcal{D}_j)$. This inequality replaces the linearization cut in traditional outer-approximation methods. We design two algorithms depending on the form of functions $g^j(\boldsymbol{x})$ in the description of $(\mathcal{E})$.

The first algorithm, as given in Algorithm 2, applies to INLPs whose functions $g^j(\boldsymbol{x})$ are integer-quasiconvex. We say that a function $g(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is *integer-quasiconvex* if $\mathrm{conv}(S^\alpha_{\mathbb{Z}}(g)) \cap \mathbb{Z}^n = S^\alpha_{\mathbb{Z}}(g)$ for any integer-α-level set $S^\alpha_{\mathbb{Z}}(g) := \{\boldsymbol{x} \in \mathbb{Z}^n \,|\, g(\boldsymbol{x}) \leq \alpha\}$. In words, any integer point of the convex hull of the integer-α-level set is feasible to the level set. It is easy to verify that integer-quasiconvexity generalizes quasiconvexity and integer-convexity.

Proposition 2. *Assume that all constraints $g^j(\boldsymbol{x})$ of $(\mathcal{E})$ are integer-quasiconvex. Algorithm 2 converges to an optimal solution of $(\mathcal{E})$ or proves that it is infeasible in a finite number of iterations.* □

Algorithm 2 DD-ECP algorithm for integer-quasiconvex INLP

Initialize $\mathcal{K} \leftarrow \emptyset$, $i \leftarrow 1$
Solve $(\mathcal{E}^D(\mathcal{K}))$
if $(\mathcal{E}^D(\mathcal{K}))$ is infeasible **then**
    Return infeasibility for $(\mathcal{E})$
else
    Find an optimal solution $\bar{\boldsymbol{x}}^i$ of $(\mathcal{E}^D(\mathcal{K}))$
end if
if $g^j(\bar{\boldsymbol{x}}^i) \leq 0$ for all $j \in J$ **then**
    Return $\bar{\boldsymbol{x}}^i$ as an optimal solution of $(\mathcal{E})$
else
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\bar{\boldsymbol{x}}^i\}$, $i \leftarrow i+1$, go to 2
end if

The second algorithm, as given in Algorithm 3, applies to general INLPs, i.e., those whose functions $g^j(\boldsymbol{x})$ are not integer-quasiconvex. In view of Algorithm 3, after the addition of $\bar{\boldsymbol{x}}^i$ to $\mathcal{K}$, $Flag$ is updated as follows. If a new DD inequality $h_j(\bar{\boldsymbol{x}}^i, \boldsymbol{x}) \leq 0$ that is violated at $\bar{\boldsymbol{x}}^i$ is added to $(\mathcal{E}^D(\mathcal{K}))$ for some $j \in \bar{J}(\bar{\boldsymbol{x}}^i)$, we assign $Flag \leftarrow False$, otherwise we assign $Flag \leftarrow True$. In words, if the optimal solution of the current ILP relaxation $(\mathcal{E}^D(\mathcal{K}))$ cannot be cut off with respect to any individual constraints of $(\mathcal{E})$, the algorithm stops and returns the current objective value as the best possible bound that can be obtained via individual constraint considerations. Otherwise, the optimal solution is separated from the feasible region of $(\mathcal{E}^D(\mathcal{K}))$ through DD inequalities and the steps are repeated. These arguments imply the following convergence results for the DD-ECP algorithm.

Algorithm 3 DD-ECP algorithm for non-integer-quasiconvex INLP

Initialize $\mathcal{K} \leftarrow \emptyset$, $i \leftarrow 1$, $Flag \leftarrow False$
Solve $(\mathcal{E}^D(\mathcal{K}))$
if $(\mathcal{E}^D(\mathcal{K}))$ is infeasible **then**
    Return infeasibility for $(\mathcal{E})$
else
    Find an optimal solution $\bar{\boldsymbol{x}}^i$ of $(\mathcal{E}^D(\mathcal{K}))$
end if
if $Flag = True$ **then**
    Return $\boldsymbol{c}^\intercal \bar{\boldsymbol{x}}^i$ as a dual bound for $(\mathcal{E})$
else
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\bar{\boldsymbol{x}}^i\}$, $i \leftarrow i+1$, update $Flag$, go to 2
end if

Proposition 3. *Algorithm 3 converges to* $p^* = \{\boldsymbol{c}^\intercal \boldsymbol{x} \,|\, \boldsymbol{x} \in \cap_{j \in J} \operatorname{conv}(\mathcal{G}^j_{\mathbb{Z}}), \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}] \cap \mathbb{Z}^n\}$ *or proves that it is infeasible in a finite number of iterations.* □

The main difference between Algorithms 2 and 3 is that the latter is not guaranteed to converge to an optimal solution of a general INLP. Nevertheless, Algorithm 3 always generates a linear outer-approximation for $(\mathcal{E})$ at termination, and hence its objective value serves as a dual bound. It turns out that these bounds are often tighter than those obtained from typical convexification techniques such as factorable relaxations where each *complicated* term at constraints is developed and decomposed separately. In contrast, the DD cuts are generated with respect to each individual constraint with all of its complicated terms, which results in capturing deeper interactions between variables on their domain and yielding stronger relaxations. Other advantages of using DD-ECP technique over the traditional outer-approximation algorithms are summarized as follows. The DD cuts are always valid for the feasible region of INLP even if the constraint functions are non-convex, providing an advantage over the traditional outer-approximation schemes that rely on validity of linearization cuts for convex functions. Unlike the linearization cuts, the DD cuts do not require any assumption on the smoothness of the functions as they are derivative-free. The DD cuts are structurally stronger than linearization cuts, as the strength of the linearization cuts depends on the distance of the point to be separated from the region defined by the constraint, whereas the DD cuts are independent of this distance and are derived with respect to the convex hull of the region defined by the constraint. Furthermore, the linearization cuts are always valid for the convex hull of the continuous points satisfied by the constraint, whereas the DD cuts are valid for the convex hull of the discrete points satisfied by the constraint. This leads to the derivation of DD inequalities that can cut through the continuous region defined by the constraint.

7 Computational Results

In this section, we showcase the potential of the DD-ECP algorithm by presenting computational results for a class of INLP that appears in pricing problems in marketing applications. We refer the reader to [6] for more comprehensive experiments on different benchmark and synthetic problem instances, as well as detailed algorithmic settings. The pricing problem is defined as follows

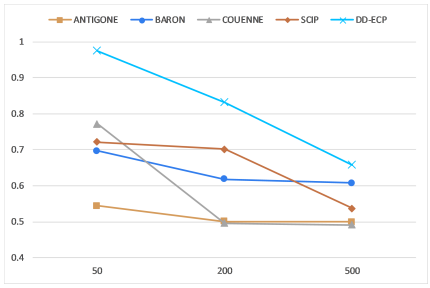
$$\min \quad \sum_{i=1}^{n} c_i x_i \tag{4a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} a_i^j x_i e^{-x_i^{k_i^j}} \geq b_j, \qquad \forall j \in J \tag{4b}$$

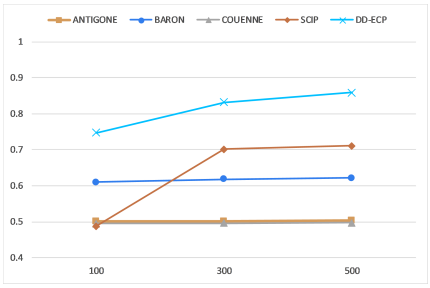
$$\boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}] \cap \mathbb{Z}^n, \tag{4c}$$

where $\boldsymbol{x}$ represents the price vector for the set of n products. The objective function is to set a nonnegative weighted sum of prices low for a firm that enters a competitive market to gain customer attention and market share. Mimicking rounded price values, the price targets are selected from integers within a range, and then rescaled to achieve the desired numeral precision. Constraint (4b) represents the profit satisfaction criterion in which the sum of profit functions over all products at each market district j must exceed a minimum profit margin b_j.

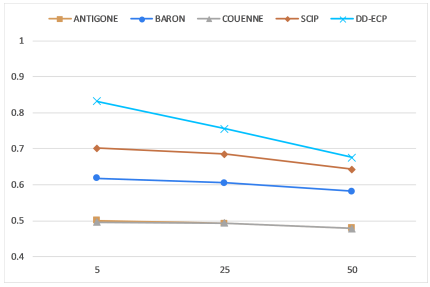
Since constraints (4b) are nonconvex, we use Algorithm 3. We evaluate the performance of our method by comparing the bounds obtained from the DD-ECP algorithm with those obtained from the state-of-the-art global solvers ANTIGONE, BARON, COUENNE and SCIP. We set the default problem size $n = 200$, constraint number $|J| = 5$, variables' bound $[\boldsymbol{l}, \boldsymbol{u}] = [0, 10]$, time limit $t = 300$ seconds. To investigate the marginal impact of these factors on the solution performance, we consider three categories of different values for each factor as reported on the horizontal axis in Figures 1a –1d. The vertical axis in these figures show the *absolute* gap improvement rate for different solvers, which is computed as the average of absolute gap closure for each algorithm along all randomly generated instances of each category. As evidenced in these figures, the DD-ECP algorithm outperforms all solvers across different categories and most of varying parameter values.



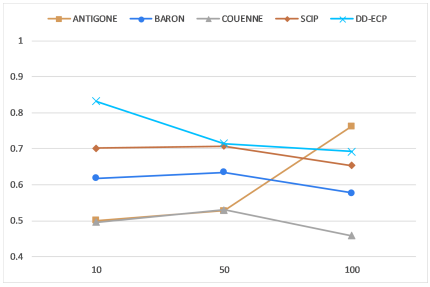
(a) Gap closure vs problem size



(b) Gap closure vs time limit



(c) Gap closure vs constraint number



(d) Gap closure vs variable bound

Figure 1. DD-ECP performance for the pricing problem

References

[1] D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. Hooker. Optimization bounds from binary decision diagrams. *INFORMS Journal on Computing*, 26:253–268, 2013.

[2] D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. Hooker. *Decision Diagrams for Optimization*. Springer International Publishing, 2016.

[3] P. Bonami, M. Kilinç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming. The IMA Volumes in Mathematics and its Applications*, volume 154. Springer, 2012.

[4] A. A. Ciré and W. J. van Hoeve. Multivalued decision diagrams for sequencing problems. *Operations Research*, 61:1411–1428, 2013.

[5] Danial Davarnia. Strong relaxations for continuous nonlinear programs based on decision diagrams. *Operations Research Letters*, 49:239–245, 2021.

[6] Danial Davarnia and Willem-Jan van Hoeve. Outer approximation for integer nonlinear programs via decision diagrams. *Mathematical Programming*, 2020.

[7] J. Kronqvist, D. Bernal, A. Lundell, and I. Grossmann. A review and comparison of solvers for convex MINLP. *Optimization and Engineering*, 20:397–455, 2017.

[8] H. Salemi and D. Davarnia. On the structure of decision diagram-representable mixed integer programs with application to unit commitment. http://www.optimization-online.org/DB_HTML/2021/01/8234.html, 2021.

[9] R. St-Aubin, J. Hoey, and C. Boutilier. Approximation policy construction using decision diagrams. In *Proceedings of Conference on Neural Information Processing Systems*, pages 1089–1095, Nantes, France, 2000.

[10] C. Tjandraatmadja and W.-J. van Hoeve. Target cuts from relaxed decision diagrams. *INFORMS Journal on Computing*, 31:285–301, 2019.

Research Highlight: A Visual Summary of the Boxed Line Method: A Criterion Space Method for Biobjective Mixed Integer Programming

by TYLER PERINI, GEORGIA INSTITUTE OF TECHNOLOGY, *perinita@gatech.edu*, NATASHIA BOLAND, DIEGO PECIN, AND MARTIN SAVELSBERGH

Despite recent interest in multiobjective integer programming, few algorithms exist for solving biobjective mixed integer programs. We present such an algorithm: the Boxed Line Method. For one of its variants, we prove that the number of single-objective integer programs solved is bounded by a linear function of the number of nondominated line segments in the nondominated frontier; this is the first such complexity result. An extensive computational study demonstrates that the Boxed Line Method is also efficient in practice, and that it outperforms previously published algorithms on a diverse set of instances.

Biobjective optimization problems with discrete decision variables arise in many fields, including scheduling [6], geographic information systems [7], facility location [5], health care [9], and many more [11]. In contrast to single objective optimization, the goal in biobjective (and, more generally, multiobjective) optimization is to generate a *set* of solutions that induces the *nondominated frontier (NDF)*, also known as the *Pareto front*. The NDF is the set of *nondominated points (NDPs)*: an NDP is a vector of objective values evaluated at a feasible solution with the property that there exists no other feasible solution that is at least as good in all objective values and is better in one or more of them. There has been enormous interest in these problems from the evolutionary algorithms community; see, for example, the surveys of [3, 4, 12].

Biobjective mixed integer linear programs (BOMIPs), unlike pure integer programs, have only recently received vigorous interest, in part due to their additional numerical challenges. BOMIP frontiers have a complex structure. The frontier can contain closed, open, and half-open line segments, as well as isolated points; see, for example, the NDF illustrated in Figure 2. Numerical tolerances, and how these are used within an algorithm, can significantly affect the representation of the NDF it produces.

Recently, criterion space methods, in which the search for the NDF operates over the space of the vector of objective function values, known as the *criterion space*, have emerged. Such methods have the advantage of being able to exploit advances in single-objective solver software, since these methods repeatedly solve single-objective problems, both linear programs (LPs) and mixed integer linear programs (which we will generically refer to as *IPs*), treating the single-objective solver as a "black box". Single-objective problems, either LP or IP, are the main "workhorses" of these algorithms, which differ mainly in the structure and number of such problems that need to be solved before the NDF is completely identified.

This paper makes the following key contributions.

1. We propose a new criterion space search method for solving the BOMIP: the Boxed Line Method. The method is

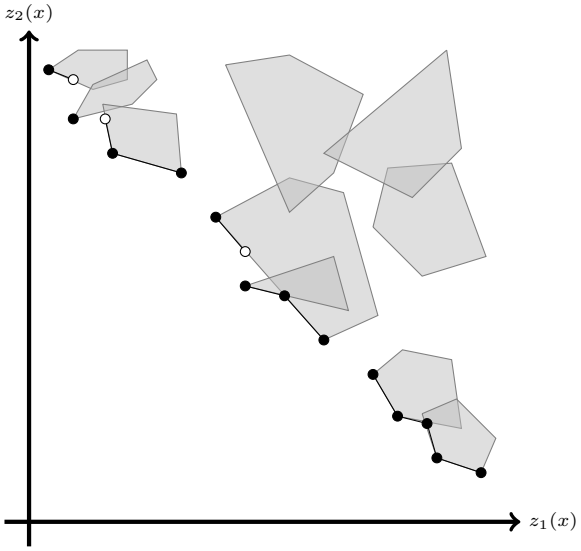


Figure 2. The nondominated frontier of a BOMIP where nondominated line segments are darkened.

designed to generalize the *Balanced Box Method (BBM)* [1], which is a computationally effective method for pure integer BOIP. The resulting algorithm is amenable to analysis (discussed next) and produces a parsimonious description of the NDF.

2. The algorithm has two variants that permit analysis of the number of single-objective IPs that they require to be solved: a basic, iterative version and a recursive version. For both variants, we provide upper bounds on the number of single-objective IPs required to produce the NDF. These are the first analytic results of this type for mixed integer multiobjective problems.
3. We design an enhancement that takes advantage of a property of the NDF encountered in many BOMILP instances, which can provide a significant improvement in runtime.
4. The benchmark instances originally proposed by [8], which have been used to test recent methods, have two primary weaknesses: (1) sensitivity to numerical tolerances and (2) redundancy in structure of the NDF which can bias comparisons of algorithms if one of the algorithms is designed to exploit this structure. We propose a new approach to creating instances in a way that facilitates validation of BOMIP algorithms by producing instances for which the frontiers are known precisely, *a priori*. It also supplements the existing suite of test instances by providing instances having different structural characteristics.
5. We provide computational results that demonstrate the relative strengths and weaknesses of the Boxed Line Method variants on two classes of the instances. The results are compared with the εTCM [10].

1 Algorithm Highlights

We define the *biobjective mixed integer linear program (BOMIP)* as

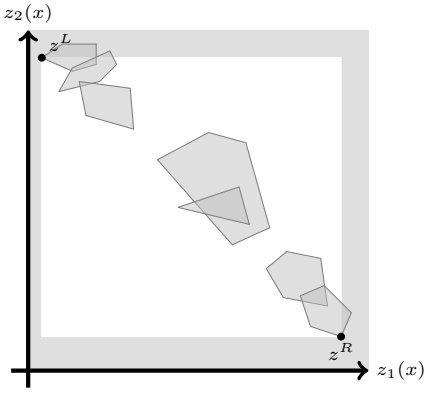
$$\min_{x \in \mathcal{X}} \{z(x) := (z_1(x), z_2(x))\} \tag{1}$$

where $z_1(x), z_2(x)$ are linear in x and the feasible region is given by $\mathcal{X} \subseteq \mathbb{Z}^{n_I} \times \mathbb{R}^{n_C}$.

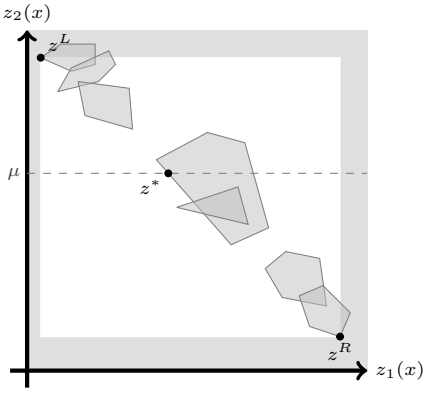
The basic approach of the BLM, similar to other criterion space algorithms, is to decompose the criterion space into a series of rectangular regions called "boxes" that are stored in a queue. Each box is processed in the same way to find nondominated line segments (NLSs) in the NDF, until the queue is empty. The algorithm is organized with an *outer loop*, which handles boxes in the queue as well as updates the NDF, and an *inner loop*, which processes each box to discover a NLS.

We introduce the outer loop at a high level in Figure 3. The white regions represent the boxes used to decompose the criterion space. Figure 3(a) shows a box chosen from the queue. The first step of processing a box is to split a box in half, horizontally, by line $z_2(x) = \mu$, as shown in Figure 3(b). By using lexicographic optimization, the leftmost NDP z^* that satisfies $z_2^* \leq \mu$ is computed. When the NDP satisfies the inequality strictly, then the algorithm avoids the inner loop and reduces to the balanced box method [1].

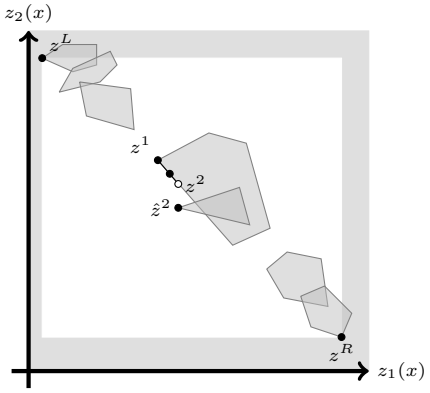
The more interesting case is when $z_2^* = \mu$. As seen in Figure 3(b), then z^* belongs to a NLS. The inner loop is called to generate the line segment from the integer solution (a.k.a., the *slice*) containing z^* and reduce this line segment to the



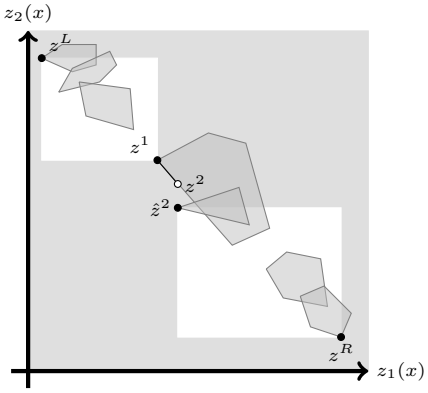
(a) Suppose the box $B(z^L, z^R)$ is chosen from the queue.



(b) Minimizing $z_1(x)$ then $z_2(x)$ below the split line μ yields z^* where $z_2^* \leq \mu$. Here we show where $z_2^* = \mu$.



(c) The Inner Loop returns the nondominated line segment containing z^* (bold). Endpoints z^1 and z^2 are indicated as open or closed. Also returned is the NDP that dominates each open endpoint, e.g. $\hat{z}^2$ above.



(d) The approximation of the NDF is updated with NLS, and two new boxes are added to the queue, e.g., $B(z^L, z^1)$ and $B(\hat{z}^2, z^R)$.

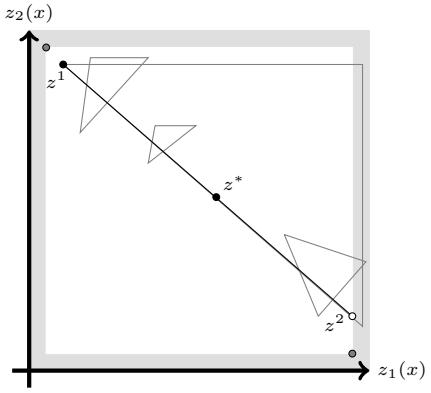
Figure 3. Outer loop procedure for the Boxed Line Method when $z_2^* = \mu$.

nondominated portion. We give the highlights of the inner loop later; the NLS resulting from the inner loop is shown in Figure 3(c).

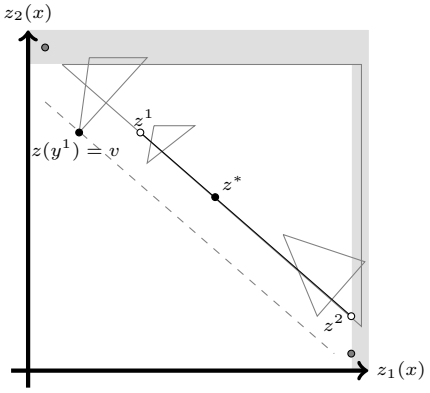
In the final steps of the outer loop, the output from the inner loop is used to update the NDF and the queue. Any found NLS are added to the NDF. The NLS and possibly other NDPs are used to define boxes that capture the remaining unexplored areas of the criterion space, as shown in 3(d). At most two boxes are added to the queue, where their summed area is at most half of the original box, and the next iteration of the outer loop begins.

Next, we introduce the basic inner loop. The inner loop takes as input the NDP z^* and returns as output the NLS containing it. An example of this procedure is shown in Figure 4. The line segment of the slice that contains z^* can be computed simply by a restricted dichotomic approach. As shown in Figure 4(a), this line segment is an overapproximation of the nondominated portion. The inner loop uses weighted sum scalarization with the gradient of the line segment to identify new NDPs that dominate a portion of the line segment. Using that new NDP, the endpoint of the line segment is updated. This iterative search and update process is shown in Figure 4(b)-(d). Once the line segment is correctly reduced to the NLS, then the weighted sum scalarization will not find any new NDPs. The inner loop returns the endpoints of the NLS along with new NDPs found.

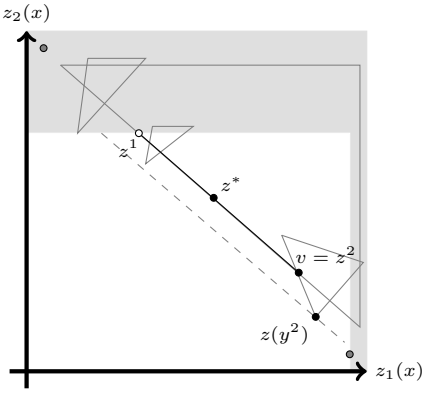
Additionally, we designed a recursive variant of the inner loop. The recursion occurs when a NDP is found to dominate a portion of the candidate line segment. At this point, the inner loop is called on the new NDP to determine the NLS containing it. Once completed, a NLS or set of NLSs is returned to the parent call. An exhibit of the recursive inner loop is illustrated in Figure 5. This recursive design provides tighter worst-case bounds and, on certain instances, better computational performance.



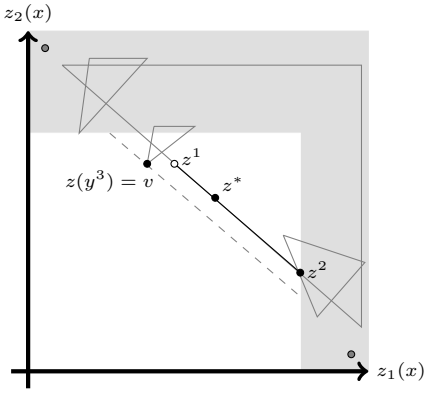
(a) The darkened line is the line segment from the slice containing z^*. This is the initial overapproximation of the NLS.



(b) Solving weighted sum scalarized IP (within the white region bounded by the previous z^1,z^2) identifies $z(y^1)$. Endpoint z^1 is updated accordingly, so it is now open.



(c) Solving the updated IP again (within the white region bounded by the previous z^1,z^2) identifies $z(y^2)$. The new endpoint z^2 is computed, updated, and indicated as closed.



(d) Solving the updated IP again yields $z(y^3)$. Update z^1, and one final IP solve confirms that $L(z^1, z^2)$ is nondominated.

Figure 4. The basic inner loop takes as input NDPs z^L, z^R, and z^*. The output is the (maximal) NLS containing z^*, $L(z^1, z^2)$, and the NDP that dominates any open endpoint.

2 Complexity Results

We prove worst-case upper bounds on the number of IPs solved in order to generate the entire NDF as a function of the number of NLSs in the NDF. For $n \geq 1$ NLS, let $\ell(n)$ be the *worst-case* number of lexicographic IPs solved by completion of BLM, and let $s(n)$ be the number of single-objective optimization IPs (e.g., scalarized IPs) solved by the basic BLM. Similarly define $\ell_R(n)$ and $s_R(n)$ for the recursive BLM.

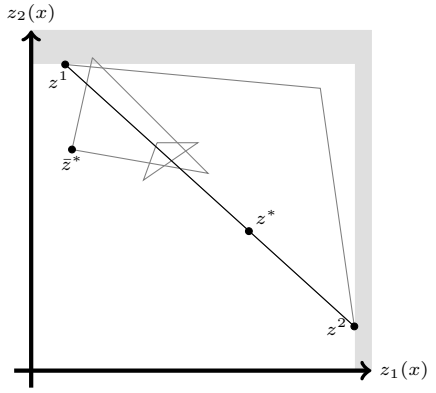
- For the outer loop, $\ell(n) = \ell_R(n) = 3n + 2$ for given $n \geq 1$.
- For the basic inner loop, $s(n) = \frac{n(n+1)}{2} + 2(n-1)$ for given $n \geq 1$.
- For the basic inner loop, $s_R(n) = 2n - 1$ for given $n \geq 1$.

Therefore, in terms of total IPs solved, the basic BLM has a quadratic bound on the worst-case number of IPs, and the recursive BLM has a linear bound.

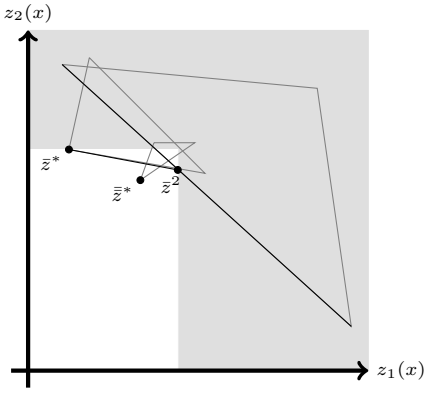
3 Computational Study

Notably, this paper challenged the status quo for comparing BOMIP algorithms. The first algorithm of this class was the Triangle Splitting Algorithm [2], which uses a relative tolerance in its computational study and therefore is unreasonable to compare more accurate algorithms that use an absolute tolerance. Secondly, there are structural weaknesses of the class of instances that have been historically used to measure the performance of BOMIP algorithms. These instances, modified from [8], all share similarly structured frontiers which result in (1) potential bias for designing and comparing algorithms and (2) in this particular case, great sensitivity to tolerances.

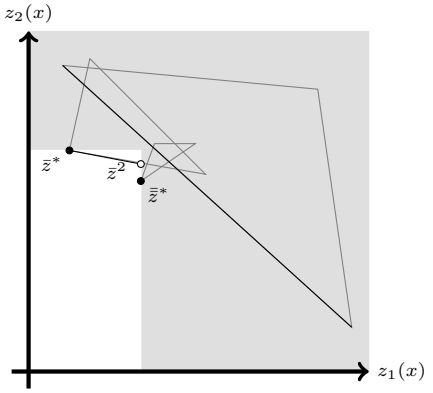
This paper provides a thorough computational study on two classes of instances comparing variants of BLM as well as



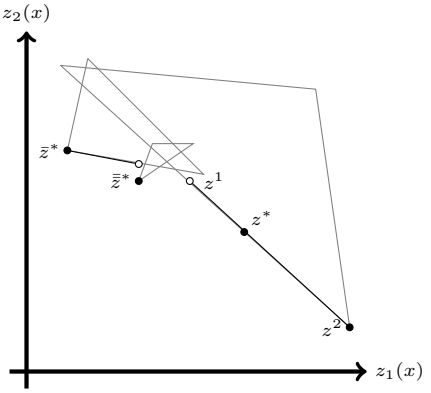
(a) Level 0: The original NDP is z^*, and initial approximation of the line is shown in bold. Solving a weighted sum scalarized IP over the white region identifies NDP $\bar{z}^*$.



(b) Level 1: The line approximated for $\bar{z}^*$ is trimmed by its intersection with L_0, resulting in $L_1 = L(\bar{z}^*, \bar{z}^2)$ with $\bar{z}^2$ closed. NDP $\bar{\bar{z}}^*$ solves the next scalarized IP.
Level 2: Returns isolated NDP $\bar{\bar{z}}^*$.



(c) Level 1: Update endpoint $\bar{z}^2$, which is now open. One more scalarized IP solve over the white region identifies that $L(\bar{z}^*, \bar{z}^2)$ is nondominated.



(d) Level 0: Update endpoint z^1 with respect to $\bar{\bar{z}}^*$, so z^1 is now open. Solving one final scalarized IP confirms that $L(z^1, z^2)$ is nondominated.

Figure 5. The recursive inner loop applied to NDP $\bar{z}^*$ returns nondominated line segments $L(z^1, z^2)$ and $L(\bar{z}^*, \bar{z}^2)$ and the isolated NDP $\bar{\bar{z}}^*$. We use level 0 to represent the base call and say that level i calls level $i + 1$.

an existing algorithm for BOMIPs. In addition to the historical instances used for previous BOMIP algorithms [8], we proposed a new class of structured instances that overcome the sensitivity issues and introduces a very different structure of frontiers.

Given two variants of BLM, the recursive variant performs better on the generated instances, and another variant performs better on the historical instances. Each variant, on its respective set of instances, also outperforms the ε-Tabu Constraint Method on the largest instances of each class.

References

[1] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A criterion space search algorithm for biobjective integer programming: the balanced box method. *INFORMS Journal on Computing*, 27(4):735–754, 2015.

[2] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A criterion space search algorithm for biobjective mixed integer programming: the triangle splitting method. *INFORMS Journal on Computing*, 27(4):597–618, 2015.

[3] Carlos A Coello. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys (CSUR)*, 32(2):109–143, 2000.

[4] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, New York, 2001.

[5] Reza Zanjirani Farahani, Maryam SteadieSeifi, and Nasrin Asgari. Multiple criteria facility location problems: a survey. *Applied Mathematical Modelling*, 34(7):1689–1709, 2010.

[6] Deming Lei. Multi-objective production scheduling: a survey. *The International Journal of Advanced Manufacturing Technology*, 43(9-10):926–938, 2009.

[7] Jacek Malczewski. GIS-based multicriteria decision analysis: a survey of the literature. *International Journal of Geographical Information Science*, 20(7):703–726, 2006.

[8] G Mavrotas and D Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107:530–541, 1998.

[9] Abdur Rais and Ana Viana. Operations research in healthcare: a survey. *International Transactions in Operational Research*, 18(1):1–31, 2011.

[10] Banu Soylu and Gazi Bilal Yıldız. An exact algorithm for biobjective mixed integer linear programming problems. *Computers & Operations Research*, 72:204–213, 2016.

[11] DJ White. A bibliography on the applications of mathematical programming multiple-objective methods. *Journal of the Operational Research Society*, 41(8):669–691, 1990.

[12] Aimin Zhou, Bo Yang Qu, Hui Li, Shi Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: a survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

Research Highlight: A Combinatorial Cut-and-Lift Procedure with an Application to 0-1 Chance Constraints

by MARGARITA P. CASTRO, *mpcastro@mie.utoronto.ca*, ANDRE A. CIRE, *andre.cire@rotman.utoronto.ca*, AND J. CHRISTOPHER BECK, *jcb@mie.utoronto.ca*, UNIVERSITY OF TORONTO

We would like to thank the INFORMS Computing Society student paper award committee members, Claudia d'Ambrosio, Georgina Hall, and Ruiwei Jiang, for the honor of being awarded runner-up in the ICS student Paper Award competition. We would also like to thank the newsletter editor, Yongjia Song, for his invitation to share our research with the ICS community. This is a summary of our joint work [12].

1 Introduction

Cutting plane methodologies have played a key role in the theoretical and computational development of mathematical programming [10, 19]. This paper studies a cut generation procedure and lifting approach for binary optimization problems of the form

$$\max_{\boldsymbol{x} \in X \subseteq \{0,1\}^n} \boldsymbol{c}^\top \boldsymbol{x}, \tag{BP}$$

where the feasible set X is arbitrary, e.g., possibly represented by a conjunction of linear and/or non-linear constraints. Our methodologies consist of exploiting *network structure* via a binary decision diagram (BDD) embedding of X. A BDD is a graphical model that represents solutions as paths in a directed acyclic graph, which can be viewed as a network-flow reformulation of X. Several BDD encodings have already been investigated for linear and non-linear problems [6, 9, 18] and are used to exploit submodularity [7] or more general combinatorial structure [8].

We propose a sequential lifting procedure applicable to any initial inequality (e.g., given by another cutting-plane technique). The lifting algorithm uses 0-1 disjunctions derived from a BDD representation of X to rotate inequalities while maintaining their validity. We show that each step of our sequential lifting, when applicable, increases the dimension of the face by at least one, and we establish conditions for which the inequality becomes facet-defining. We also relate our procedure with lifting techniques from disjunctive programming [5], showing that our approach generalizes well-known lifting procedures for 0-1 inequalities [4, 15, 20].

For our cut generation approach, we propose a new linear formulation of the BDD polytope based on capacitated flows, which leads to an alternative cut generation linear program (CGLP) for separating infeasible points from X. We show that the set of cuts derived from this model defines the convex hull of the solutions encoded by the BDD. Moreover, in contrast to recent cutting-plane algorithms based on BDDs [14, 22], our CGLP does not require any additional information about X, such as interior points or normalization constraints. Finally, for practical purposes, we present a weaker but computationally faster alternative that solves a max-flow problem over the BDD to generate valid inequalities.

For optimization problems where a BDD for X may be exponentially large in n, our lifting and cut procedures remain valid when considering instead a limited-size *relaxed* BDD for BP, i.e., where the BDD encodes a superset of X. Several efficient methods exist to build relaxed BDDs, such as only considering a subset of the problem constraints [8]. This approach is similar in spirit, e.g., to when a linear relaxation is used to lift cover inequalities of a single knapsack constraint [4]. Nonetheless, here we exploit the discrete relaxation provided by the BDD as opposed to a continuous relaxation, which captures some of the combinatorial structure of the problem.

As a case study, we apply our combinatorial cut-and-lift procedure to a class of 0-1 chance-constrained problems. Chance constraints are common in stochastic optimization to model uncertainty or enforce robustness [2, 21, 13]. In particular, we focus on binary problems with normally distributed chance constraints:

$$\max_{\boldsymbol{x}\in\{0,1\}^n} \boldsymbol{c}^\top \boldsymbol{x} \tag{CC}$$
$$\text{s.t. } \mathbb{P}(\boldsymbol{a}_j^\top \boldsymbol{x} \le b_j) \ge \epsilon_j, \qquad \forall j \in \{1,...,m\},$$

where, for each j, $\boldsymbol{a}_j \in \mathbb{R}^n$ are random variables with a joint normal distribution and ϵ_j is a threshold probability. Each inequality can be rewritten as a second-order cone (SOC) constraint [23], which are amenable to commercial solvers, e.g., CPLEX [16].

We investigate problems with multiple SOC inequalities, each reformulated with an appropriate BDD encoding. We experiment on the knapsack chance-constraint benchmark [3, 17] and over 270 randomly generated instances with joint-distributed chance constraints, incorporating both our general cutting and lifting approaches into CPLEX. We show that our combinatorial cut-and-lift procedure achieves a 52.2% average root gap reduction, has comparable average solving time, and solves 17 more instances on the knapsack benchmark when compared to existing cut-and-lift methodologies [3]. Similarly, our procedure outperforms CPLEX on the random dataset by achieving a 35.3% average root gap reduction, solving 31 more instances (168 vs. 137), and reducing the mean run-time threefold.

2 Background

This section introduces the notation used throughout this work and the background material on decision diagrams for optimization. For convenience, we assume $n \ge 1$ and let $I := \{1,...,n\}$ represent the component indices of any point $\boldsymbol{x}$ in an n-dimensional set.

Binary Decision Diagrams. A BDD $\mathcal{B}$ is an extended representation of a set $X_{\mathcal{B}} \subseteq \{0,1\}^n$ as a network. Specifically, $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ is a layered directed acyclic graph with node set $\mathcal{N}$ and arc set $\mathcal{A}$. The node set $\mathcal{N}$ is partitioned into $n+1$ *layers* $\mathcal{N} = (\mathcal{N}_1,...,\mathcal{N}_{n+1})$. The first and last layers are the singletons $\mathcal{N}_1 = \{\mathbf{r}\}$ and $\mathcal{N}_{n+1} = \{\mathbf{t}\}$, respectively, where $\mathbf{r}$ is the root node and $\mathbf{t}$ is the terminal node. An arc $a = (u,u') \in \mathcal{A}$ has a source node $s(a) = u$ and a target node $t(a) = u'$ in consecutive layers, i.e., $u' \in \mathcal{N}_{i+1}$ whenever $u \in \mathcal{N}_i$ for $i \in I$.

The points of $X_{\mathcal{B}}$ are mapped to paths in the network, as follows. With each arc $a \in \mathcal{A}$ we associate a value $v_a \in \{0,1\}$, where a node $u \in \mathcal{N}$ has at most one arc of each value emanating from it. Given an arc-specified $\mathbf{r}-\mathbf{t}$ path $p = (a_1,...,a_n)$ with $s(a_1) = \mathbf{r}$ and $t(a_n) = \mathbf{t}$, we let $\boldsymbol{x}^p := (v_{a_1}, v_{a_2}, \ldots, v_{a_n}) \in \{0,1\}^n$ be the n-dimensional point encoded by path p. Thus, if $\mathcal{P}$ is the set of all $\mathbf{r}-\mathbf{t}$ paths in $\mathcal{B}$, the set of points represented by the BDD is $X_{\mathcal{B}} = \bigcup_{p\in\mathcal{P}}\{\boldsymbol{x}^p\}$.

A BDD $\mathcal{B}$ is *exact* for set $X \subseteq \{0,1\}^n$ when $X = X_{\mathcal{B}}$, i.e., there is a one-to-one relationship between the points in X and the $\mathbf{r}-\mathbf{t}$ paths in $\mathcal{B}$. Alternatively, $\mathcal{B}$ is *relaxed* when $X \subseteq X_{\mathcal{B}}$, i.e., every point in X maps to a path in $\mathcal{B}$ but the converse is not necessarily true.

Example 1. *Consider $X = \{\boldsymbol{x} \in \{0,1\}^4 : 7x_1 + 5x_2 + 4x_3 + x_4 \le 8\}$. Figure 6 illustrates two exact BDDs for X: $\mathcal{B}_1$ on the left-hand side and $\mathcal{B}_2$ on the right-hand side. Dashed and solid arcs have a value of 0 and 1, respectively. Each point $\boldsymbol{x} \in X$ is represented by a path in $\mathcal{B}_1$ and $\mathcal{B}_2$. For example, $\boldsymbol{x} = (1,0,0,1) \in X$ is encoded by the path $((\mathbf{r},u_2), (u_2,u_4), (u_4,u_5), (u_5,\mathbf{t}))$ in $\mathcal{B}_1$, and by the path $((\mathbf{r},u'_2), (u'_2,u'_5), (u'_5,u'_6), (u'_6,\mathbf{t}))$ in $\mathcal{B}_2$.* □

A BDD $\mathcal{B}$ is *reduced* if it is the smallest network (with respect to number of nodes) that represents the set $X_{\mathcal{B}}$. There exists a unique reduced BDD for a given ordering of the indices I. Furthermore, given any $\mathcal{B}$ and an ordering, we can obtain its associated reduced BDD in polynomial time in the size of $\mathcal{B}$ [11]. For instance, $\mathcal{B}_1$ in Figure 6 is reduced and is obtained by merging nodes u'_4 and u'_5 from $\mathcal{B}_2$.

Several exact and relaxed BDD construction mechanisms are available for general and specialized discrete optimization problems [8, 7]. These techniques are based on either reformulating the problem as a dynamic program, where $\mathcal{B}$ represents an underlying state-transition graph, or by separating infeasible paths of a relaxed BDD. It is often the case that BDDs can be exponentially smaller than enumerating $X_{\mathcal{B}}$ explicitly [8]. If exact BDDs are too large, relaxed BDDs can be built for X by either imposing a limit on the number of nodes (e.g., a polynomial in the problem input) or by considering only a subset of the problem constraints.

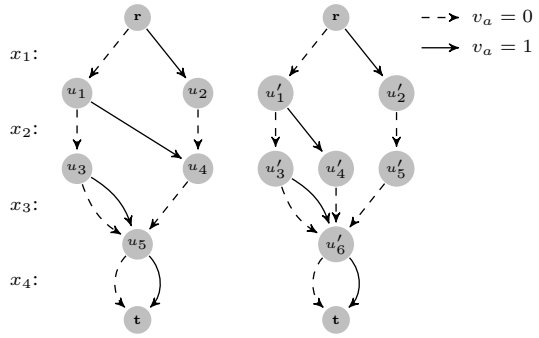


Figure 6. Two BDDs $\mathcal{B}_1$ (left-hand side) and $\mathcal{B}_2$ (right-hand side) with $X_{\mathcal{B}_1} = X_{\mathcal{B}_2}$. $\mathcal{B}_1$ is reduced.

3 Combinatorial Lifting

We now present our combinatorial lifting procedure and develop its structural properties. Throughout this section, we assume that, for a given $X \subseteq \{0,1\}^n$, (a) $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ is a valid inequality that supports $\mathrm{conv}(X)$; (b) $\mathcal{B}$ is an exact BDD for X, i.e., $X_{\mathcal{B}} = X$; and (c) For any $i \in I$, there exists $\boldsymbol{x}, \boldsymbol{x}' \in X$ such that $x_i = 0$ and $x'_i = 1$. Assumption (a) is a common lifting condition that is satisfied by setting $\pi_0 := \max_{\boldsymbol{x} \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x}\}$. This, in turn, can be enforced in linear time in the size of $\mathcal{B}$ (see §3.2). Assumption (b) is needed for our theoretical results but it can be relaxed in practice (see §3). For (c), we can soundly remove any i-th component not satisfying the assumption, adjusting n accordingly.

Our goal is to lift $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ and better represent $\mathrm{conv}(X)$ by exploiting the network structure of $\mathcal{B}$. The resulting cuts are valid for any subset $X' \subseteq X$; e.g., when $\mathcal{B}$ (and hence X) is a relaxation of some feasible set.

3.1 Disjunctive Slack Lifting

The core element of our lifting procedure is what we denote by *disjunctive slack vector* (or *d-slack*). The i-th component of the d-slack indicates the change in the maximum values of the left-hand side of $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ when varying x_i.

Definition 1. *The disjunctive slack vector $\boldsymbol{\lambda}(\boldsymbol{\pi})$ with respect to $\boldsymbol{\pi}$ is*

$$\lambda_i(\boldsymbol{\pi}) := \lambda_i^0(\boldsymbol{\pi}) - \lambda_i^1(\boldsymbol{\pi}), \quad \forall i \in I,$$

with $\lambda_i^0(\boldsymbol{\pi}) := \max_{\boldsymbol{x} \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x} : x_i = 0\}$ and $\lambda_i^1(\boldsymbol{\pi}) := \max_{\boldsymbol{x} \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x} : x_i = 1\}$.

We now show in Theorem 1 how to apply the d-slacks to lift $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$. In particular, the resulting inequality is valid for X (and thereby $\mathrm{conv}(X)$), the dimension of the face necessarily increases, and points separated by the original inequality are still separated after lifting. This last characteristic is important, e.g., if the input inequality $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ was derived to separate a fractional point.

Theorem 1. *Suppose $\lambda_i(\boldsymbol{\pi}) \neq 0$ for some $i \in I$. Let $\langle \boldsymbol{\pi}', \pi'_0 \rangle$ be such that*

$$\pi'_j := \begin{cases} \pi_j & \text{if } j \neq i, \\ \pi_j + \lambda_j(\boldsymbol{\pi}) & \text{otherwise,} \end{cases} \qquad \forall j \in I, \quad \textit{and}$$

$$\pi'_0 := \begin{cases} \pi_0 & \text{if } i \in S^+(\boldsymbol{\pi}), \\ \pi_0 + \lambda_i(\boldsymbol{\pi}) & \text{otherwise.} \end{cases}$$

The following properties hold:

(1) $\boldsymbol{\pi}'^\top \boldsymbol{x} \leq \pi'_0$ is valid for X.
(2) $F(\boldsymbol{\pi}) \subset F(\boldsymbol{\pi}')$ and $\dim(F(\boldsymbol{\pi}')) \geq \dim(F(\boldsymbol{\pi})) + 1$.
(3) For any $\bar{\boldsymbol{x}} \in [0,1]^n$ with $\boldsymbol{\pi}^\top \bar{\boldsymbol{x}} > \pi_0$, we have that $\boldsymbol{\pi}'^\top \bar{\boldsymbol{x}} > \pi'_0$.

3.2 Extracting D-Slacks from BDDs

Identifying d-slacks $\boldsymbol{\lambda}(\boldsymbol{\pi})$ is a non-trivial task since we are required to solve $2n$ binary optimization problems, i.e., one for each component $i \in I$ and values 0 and 1. We develop a procedure to compute d-slacks by exploiting the network representation of a BDD $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ for X. We also show that the procedure complexity is linear in the number of arcs $|\mathcal{A}|$.

Algorithm 4 Sequential Combinatorial Lifting Procedure ($\langle \boldsymbol{\pi}, \pi_0 \rangle$, $\mathcal{B}$)

1: Calculate the disjunctive slacks $\boldsymbol{\lambda}(\boldsymbol{\pi})$
2: **while** $\boldsymbol{\lambda}(\boldsymbol{\pi}) \neq \mathbf{0}$ **do**
3: Choose $i \in I$ such that $\lambda_i(\boldsymbol{\pi}) \neq 0$
4: Apply Theorem 1 to calculate $\langle \boldsymbol{\pi}', \pi_0' \rangle$
5: Set $\langle \boldsymbol{\pi}, \pi_0 \rangle = \langle \boldsymbol{\pi}', \pi_0' \rangle$
6: Recalculate $\boldsymbol{\lambda}(\boldsymbol{\pi})$
7: **end while**
8: **return** $\langle \boldsymbol{\pi}, \pi_0 \rangle$

First, for each arc $a \in \mathcal{A}$ with value $v_a \in \{0,1\}$ and source $s(a) \in \mathcal{N}_i$ for some $i \in I$, we associate a *length* of $\pi_i \cdot v_a$. Note that the longest $\mathbf{r} - \mathbf{t}$ path of $\mathcal{B}$ with respect to such lengths maximizes $\boldsymbol{\pi}^\top \boldsymbol{x}$ over X. Similarly, given the $\mathbf{r} - \mathbf{t}$ paths $\mathcal{P}$ of $\mathcal{B}$, let

$$\ell_a := \max \left\{ \sum_{k=1}^{n} \pi_k \cdot v_{a_k} : p = (a_1, \ldots, a_n) \in \mathcal{P}, a_i = a \right\}$$

be the longest-path value conditioned on all paths that include a. Because each index $i \in I$ is uniquely associated with layer $\mathcal{N}_i$, it follows that

$$\lambda_i^j(\boldsymbol{\pi}) = \max_{a \in \mathcal{A}} \{\ell_a : s(a) \in \mathcal{N}_i, v_a = j\}, \quad \forall j \in \{0,1\},$$

and the final d-slacks are obtained by the differences $\lambda_i^0(\boldsymbol{\pi}) - \lambda_i^1(\boldsymbol{\pi})$ for all i.

The lengths ℓ_a are derived by performing two longest-path computations over $\mathcal{B}$. Specifically, let $\mathcal{A}^{\text{in}}(u)$ and $\mathcal{A}^{\text{out}}(u)$ be the set of incoming and outgoing arcs of a node $u \in \mathcal{N}$, respectively. The solution of the recursion $L^{\downarrow}(\boldsymbol{\pi}, \mathbf{r}) = 0$,

$$L^{\downarrow}(\boldsymbol{\pi}, u) = \max_{a \in \mathcal{A}^{\text{in}}(u)} \{L^{\downarrow}(\boldsymbol{\pi}, s(a)) + \pi_{i-1} v_a\},$$

for all $u \in \mathcal{N}_i$, $i \in \{2, \ldots, n+1\}$, provides the longest-path value $L^{\downarrow}(\boldsymbol{\pi}, u)$ from $\mathbf{r}$ to u, while the recursion $L^{\uparrow}(\boldsymbol{\pi}, \mathbf{t}) = 0$,

$$L^{\uparrow}(\boldsymbol{\pi}, u) = \max_{a \in \mathcal{A}^{\text{out}}(u)} \{L^{\uparrow}(\boldsymbol{\pi}, t(a)) + \pi_i v_a\},$$

for all $u \in \mathcal{N}_i$, $i \in \{1, \ldots, n\}$, provides the longest-path value $L^{\uparrow}(\boldsymbol{\pi}, u)$ from u to $\mathbf{t}$. The values $L^{\downarrow}(\boldsymbol{\pi}, u)$ can be calculated via a top-down pass on $\mathcal{B}$, i.e., starting from $\mathbf{r}$ and considering one layer $\mathcal{N}_2, \ldots, \mathcal{N}_{n+1}$ at a time. Analogously, the values $L^{\uparrow}(\boldsymbol{\pi}, u)$ are obtained via a bottom-up pass on $\mathcal{B}$, i.e., starting from $\mathbf{t}$ and considering one layer $\mathcal{N}_n, \mathcal{N}_{n-1}, \ldots, \mathcal{N}_1$ at a time. For any arc $a = (s(a), t(a))$ such that $s(a) \in \mathcal{N}_i$,

$$\ell_a = L^{\downarrow}(\boldsymbol{\pi}, s(a)) + L^{\uparrow}(\boldsymbol{\pi}, t(a)) + \pi_i \cdot v_a.$$

Since each arc is traversed twice, the complexity of the procedure is $\mathcal{O}(|A|)$.

3.3 Sequential Lifting

The lifting procedure detailed in Theorem 1 can be applied sequentially to strengthen an inequality. We summarize the procedure in Algorithm 4. The choice of i in step 3 is critical to the dimension of the resulting face, as illustrated in Example 2.

Example 2. *Consider the set* $X = \{x \in \{0,1\}^3 : 5x_1 + 2x_2 + 3x_3 \leq 6\}$ *and inequality* $\boldsymbol{\pi}^\top \boldsymbol{x} = x_1 + x_2 + x_3 \leq 2$ *that supports* $\text{conv}(X)$. *We have* $\boldsymbol{\lambda}(\boldsymbol{\pi}) = (1, -1, -1)^\top$ *and the lifted inequality with respect to* $\lambda_1(\boldsymbol{\pi}) = 1$ *is* $\boldsymbol{\pi}'^\top \boldsymbol{x} = 2x_1 + x_2 + x_3 \leq 2$ *and has* $\boldsymbol{\lambda}(\boldsymbol{\pi}') = \mathbf{0}$. *The lifted inequality is not facet-defining since* $\dim(\text{conv}(X)) = 3$ *and* $\dim(F(\boldsymbol{\pi}')) = 1$.

If we instead lift $x_1 + x_2 + x_3 \leq 2$ *with respect to* $\lambda_2(\boldsymbol{\pi}) = -1$, *we get* $\boldsymbol{\pi}'^\top \boldsymbol{x} = x_1 + x_3 \leq 1$ *and* $\boldsymbol{\lambda}(\boldsymbol{\pi}') = \mathbf{0}$. *In this case, the lifted inequality is facet-defining since* $\dim(F(\boldsymbol{\pi}')) = 2$.

In order to understand the impact of the index choice, we first show in Lemma 1 a relationship between d-slacks and the dimension of the face. We later use this result to gauge when the sequential procedure leads to a facet-defining inequality.

Lemma 1. *The dimension of a face* $F(\boldsymbol{\pi})$ *satisfies* $\dim(F(\boldsymbol{\pi})) \leq |S^0(\boldsymbol{\pi})|$. *Moreover,* $|S^0(\boldsymbol{\pi})| = 0$ *if* $\dim(F(\boldsymbol{\pi})) = 0$.

Example 2 depicts a case where $|S^0(\boldsymbol{\pi})|$ increases faster than the number of affinely independent points in $F(\boldsymbol{\pi})$. In view of Lemma 1, we would like to choose i so that $|S^0(\boldsymbol{\pi})|$ increases at a slower rate, since each lifting operation increases $\dim(F(\boldsymbol{\pi}))$ by at least one according to Theorem 1-(2). We show in Theorem 2 that the slow increase of $|S^0(\boldsymbol{\pi})|$ occurs when there exists a unique slack with minimum non-zero absolute value.

Theorem 2. *Suppose there exists $i \notin S^0(\boldsymbol{\pi})$ such that $|\lambda_i(\boldsymbol{\pi})| < |\lambda_{i'}(\boldsymbol{\pi})|$ for all $i' \notin S^0(\boldsymbol{\pi})$ $(i' \neq i)$. Then, for $\langle \boldsymbol{\pi}', \pi_0' \rangle$ obtained when lifting $\langle \boldsymbol{\pi}, \pi_0 \rangle$ with respect to $\lambda_i(\boldsymbol{\pi})$, $\dim(F(\boldsymbol{\pi}')) = \dim(F(\boldsymbol{\pi})) + 1$ and $|S^0(\boldsymbol{\pi}')| = |S^0(\boldsymbol{\pi})| + 1$.*

Theorem 2 provides a simple choice rule based on picking i with the minimum absolute d-slack. It also indicates when this rule will converge to a facet-defining inequality. We formalize it in Corollary 1, derived as a direct consequence of Theorem 2.

Corollary 1. *If $\dim(F(\boldsymbol{\pi})) = |S^0(\boldsymbol{\pi})|$, the sequential lifting procedure (Algorithm 4) produces a facet-defining inequality if, at each lifting iteration except the last, the chosen $i \in I$ is such that $|\lambda_i(\boldsymbol{\pi})| < |\lambda_{i'}(\boldsymbol{\pi})|$ for all $i' \notin S^0(\boldsymbol{\pi})$ $(i' \neq i)$.*

Finally, we note that it may not be possible to achieve a facet-defining inequality. For example, all non-zero d-slacks have the same absolute value and $|S^0(\boldsymbol{\pi})|$ might increase by more than one while the dimension of $F(\boldsymbol{\pi})$ does not (see Example 2).

We remark that the algorithm above is similar in spirit to the lifting procedures for cover inequalities based on dynamic programming [24]. There is also a strong relation between our sequential lifting and the n-step lifting procedure by Perregaard and Balas [20]. We refer the reader to our paper for details [12].

4 BDD-based Cuts

While the BDD-based lifting procedure developed in §3 can enhance inequalities from any cutting-plane methodology, we now exploit similar concepts to derive new valid inequalities for X based on the network structure of $\mathcal{B}$. In particular, we design inequalities that separate points from X by only relying on the combinatorial structure encoded by $\mathcal{B}$.

4.1 BDD Polytope

Existing BDD-based cut generation procedures [14, 18, 22] rely on the network-flow formulation $\text{NF}(\mathcal{B})$ introduced by [6], described as follows:

$$
\begin{aligned}
\text{NF}(\mathcal{B}) := \{(\boldsymbol{x}; \boldsymbol{y}) \in [0,1]^n \times \mathbb{R}_+^{|\mathcal{A}|} :& \\
\sum_{a \in \mathcal{A}^{\text{out}}(u)} y_a - \sum_{a \in \mathcal{A}^{\text{in}}(u)} y_a = 0, \quad & \forall u \in \mathcal{N} \setminus \{\mathbf{r}, \mathbf{t}\}, && \text{(1a)} \\
\sum_{a \in \mathcal{A}^{\text{out}}(\mathbf{r})} y_a = \sum_{a \in \mathcal{A}^{\text{in}}(\mathbf{t})} y_a = 1, & && \text{(1b)} \\
\sum_{a \in \mathcal{A}: s(a) \in \mathcal{N}_i, v_a = 1} y_a = x_i, \quad & \forall i \in I \Big\}. && \text{(1c)}
\end{aligned}
$$

Equalities (1a) and (1b) are balance-of-flow constraints. Constraint (1c) links the arcs of $\mathcal{B}$ with $\boldsymbol{x}$. In particular, $\text{NF}(\mathcal{B})$ projected over the $\boldsymbol{x}$ variables is equivalent to the convex hull of all solutions represented by $\mathcal{B}$, i.e., $\text{proj}_{\text{x}}(\text{NF}(\mathcal{B})) = \text{conv}(X)$.

We propose an alternative formulation to $\text{NF}(\mathcal{B})$ to define our cutting-plane algorithm. The new formulation, $\text{JNF}(\mathcal{B})$, corresponds to a joint capacitated network-flow polytope. The model maintains the balance-of-flow constraints and replaces (1c) with (2a)-(2b). Both inequalities enforce a common capacity for arcs in a layer with the same value. Proposition 4 shows that the two formulations are equivalent and, thus, $\text{proj}_{\text{x}}(\text{JNF}(\mathcal{B})) = \text{conv}(X)$.

$$
\begin{aligned}
\text{JNF}(\mathcal{B}) := \{(\boldsymbol{x}; \boldsymbol{y}) \in [0,1]^n \times \mathbb{R}_+^{|\mathcal{A}|} :& && \text{(1a)}, \text{(1b)}, \\
\sum_{a \in \mathcal{A}: s(a) \in \mathcal{N}_i, v_a = 1} y_a \leq x_i, & \quad \forall i \in I, && \text{(2a)} \\
\sum_{a \in \mathcal{A}: s(a) \in \mathcal{N}_i, v_a = 0} y_a \leq 1 - x_i, & \quad \forall i \in I \ \}. && \text{(2b)}
\end{aligned}
$$

Proposition 4. *$\text{JNF}(\mathcal{B}) = \text{NF}(\mathcal{B})$.*

4.2 General BDD Flow Cuts

Our cutting-plane procedure formulates a max-flow optimization problem over JNF($\mathcal{B}$) to identify and separate points $\boldsymbol{x}' \notin \text{conv}(X)$, given by (3) below:

$$z(\mathcal{B};\boldsymbol{x}') = \max_{\substack{\boldsymbol{y}\in\mathbb{R}_+^{|\mathcal{A}|}\\ \boldsymbol{x}=\boldsymbol{x}'}} \left\{ \sum_{a\in\mathcal{A}^{\text{out}}(\mathbf{r})} y_a : (1a), (2a), (2b) \right\} \tag{3}$$

We note that (3) omits the constraint enforcing the flow to be equal to one as in (1b). We argue in Lemma 2 that the condition $z(\mathcal{B};\boldsymbol{x}') = 1$ is necessary and sufficient to check if $\boldsymbol{x}'$ belongs to $\text{conv}(X)$.

Lemma 2. *$\boldsymbol{x}' \in \text{conv}(X)$ iff $z(\mathcal{B};\boldsymbol{x}') = 1$.*

Our BDD-based CGLP uses the dual of (3) to generate valid inequalities that cut off $\boldsymbol{x}' \notin \text{conv}(X)$. Consider $\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|}$ as the dual variables associated with constraints (1a), and $\boldsymbol{\nu}, \boldsymbol{\eta} \in \mathbb{R}^n_+$ as the dual variables for (2a) and (2b), respectively. Let $w(\mathcal{B};\boldsymbol{x}')$ be the optimal solution value of the dual problem. Strong duality and Lemma 2 imply that we can identify if $\boldsymbol{x}'$ belongs to $\text{conv}(X)$ if $w(\mathcal{B};\boldsymbol{x}') = 1$. Furthermore, we can use the optimal dual solution $(\boldsymbol{\nu}^*;\boldsymbol{\eta}^*)$ to create a valid cut when $w(\mathcal{B};\boldsymbol{x}') < 1$:

$$\sum_{i\in I} x_i\nu_i^* + \sum_{i\in I}(1-x_i)\eta_i^* \geq 1. \tag{4}$$

Theorem 3 shows that the set of all cuts of the form (4) describes $\text{conv}(X)$.

Theorem 3. *Let $\Lambda(\mathcal{B})$ be the set of extreme points of the polyhedron defined by dual of $z(\mathcal{B};\boldsymbol{x}')$. Furthermore, let $P_{\mathcal{B}}$ be the set of points $\boldsymbol{x} \in [0,1]^n$ that satisfy (4) for all $(\boldsymbol{\nu};\boldsymbol{\eta}) \in \text{proj}_{\boldsymbol{\nu},\boldsymbol{\eta}}(\Lambda(\mathcal{B}))$. Then, $\text{conv}(X) = P_{\mathcal{B}}$.*

Thus, our cutting-plane procedure separates points $\boldsymbol{x}' \notin \text{conv}(X)$ by solving the dual of (3). The procedure returns a cut of the form (4) where $(\boldsymbol{\nu}^*;\boldsymbol{\eta}^*)$ is an optimal solution of $w(\mathcal{B};\boldsymbol{x}')$.

4.3 Combinatorial BDD Flow Cuts

The above cutting-plane procedure requires solving a linear program with $|\mathcal{A}|$ constraints and $|\mathcal{N}| + 2n$ variables. Obtaining $w(\mathcal{B};\boldsymbol{x}')$, thus, could be computationally expensive for instances where $\mathcal{B}$ is large. We propose now an alternative cut-generation procedure that involves a combinatorial and more efficient max-flow solution over $\mathcal{B}$.

First, we consider a relaxation of JNF($\mathcal{B}$) where the joint capacity constraints are replaced by individual constraints for each arc, i.e., a standard capacitated network flow polytope over $\mathcal{B}$:

$$\begin{aligned}
&\text{CNF}(\mathcal{B}) := \{(\boldsymbol{x};\boldsymbol{y}) \in [0,1]^n \times \mathbb{R}_+^{|\mathcal{A}|} : (1a) - (1b),\\
&y_a \leq x_i, \qquad \forall a \in \mathcal{A}, s(a) \in \mathcal{N}_i, v_a = 1, i \in I, &\text{(5a)}\\
&y_a \leq 1 - x_i, \ \forall a \in \mathcal{A}, s(a) \in \mathcal{N}_i, v_a = 0, i \in I\}. &\text{(5b)}
\end{aligned}$$

Proposition 5. *JNF($\mathcal{B}$) $\subseteq$ CNF($\mathcal{B}$). Moreover, for any integer $\boldsymbol{x} \notin \text{conv}(X)$, we have that $\boldsymbol{x} \notin \text{proj}_{\text{x}}(\text{CNF}(\mathcal{B}))$.*

Proposition 5 shows that for any integer point $\boldsymbol{x}'$, $\boldsymbol{x}' \notin X$ implies $\boldsymbol{x}' \notin \text{proj}_{\text{x}}(\text{CNF}(\mathcal{B}))$. Example 3 illustrates that, conversely, there might exist fractional points $\boldsymbol{x}' \notin \text{conv}(X)$ such that $\boldsymbol{x}' \in \text{proj}_{\text{x}}(\text{CNF}(\mathcal{B}))$, and hence CNF($\mathcal{B}$) is a weaker representation.

Example 3. *Consider $X = \{\boldsymbol{x} \in \{0,1\}^4 : 7x_1 + 5x_2 + 4x_3 + x_4 \leq 8\}$, a fractional point $\boldsymbol{x}' = (0.4, 0.6, 0.4, 1)$, and the exact BDD $\mathcal{B}_1$ in Figure 6. It is easy to see that $\boldsymbol{x}' \notin \text{conv}(X)$ since $7x'_1 + 5x'_2 + 4x'_3 + x'_4 = 8.4 \geq 8$. However, there exists a $\boldsymbol{y}' \in \mathbb{R}_+^{|\mathcal{A}|}$ such that $(\boldsymbol{x}', \boldsymbol{y}') \in$ CNF($\mathcal{B}$) with value $y_{(\mathbf{r},u_1)} = 0.6$, $y_{(\mathbf{r},u_2)} = 0.4$, $y_{(u_1,u_4)} = 0.2$, $y_{(u_1,u_3)} = 0.4$, $y_{(u_2,u_4)} = 0.4$, $y_{(u_3,u_4)} = 0.4$, $y_{(u_4,u_5)} = 0.6$, $y_{(u_5,\mathbf{t})} = 1$, and all other arcs with flow equal to zero.*

Similar to the general BDD flow cuts, we use the dual of the max-flow version of CNF($\mathcal{B}$) to identify points that do not belong to $\text{conv}(X)$. Consider $\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|}$ as the dual variables associated with constraints (1a) and $\boldsymbol{\alpha}$ the dual variables associated with constraints (5a)-(5b). Let $w^{\text{r}}(\mathcal{B};\boldsymbol{x}')$ be the optimal solution value of the dual problem. Proposition 5 implies that for any $\boldsymbol{x}' \in \text{conv}(X_{\mathcal{B}})$, $w^{\text{r}}(\mathcal{B};\boldsymbol{x}') = 1$. It follows that inequality (6) holds for any $\boldsymbol{x} \in \text{conv}(X_{\mathcal{B}})$, where $\boldsymbol{\alpha}^*$ is optimal to the dual:

$$\sum_{i\in I}\left(\sum_{\substack{a\in\mathcal{A}:v_a=1,\\ s(a)\in\mathcal{N}_i}} x_i\alpha_a^* + \sum_{\substack{a\in\mathcal{A}:v_a=0,\\ s(a)\in\mathcal{N}_i}} (1-x_i)\alpha_a^* \right) \geq 1. \tag{6}$$

Of important note is that this separation problem is a classical min-cut problem, i.e., we are searching for a maximum-capacity arc cut in the network that certifies that a point does not belong to the convex hull of X. While the resulting inequalities are not as strong as the general BDD cuts (4) , we can leverage max-flow/min-cut combinatorial algorithms to solve it more efficiently in the size of the BDD. Several algorithms are readily available to that end [1] and provide both primal and dual solutions.

Furthermore, a consequence of the design of such cuts is that their strength depends on the BDD size. That is, two BDDs $\mathcal{B}$ and $\mathcal{B}'$ encoding the same set might generate different combinatorial flow cuts because of distinct min-cut solutions. We show in Theorem 4 that the reduced BDD, which is unique, generates the tightest CNF($\mathcal{B}$) formulation and is hence critical in such a formulation. We note that a reduced BDD can be generated in polynomial time in $\mathcal{B}'$ for any $\mathcal{B}'$ representing the desired solution set [11].

Theorem 4. *Let $\mathcal{B}^r = (\mathcal{N}^r, \mathcal{A}^r)$ be the reduced version of $\mathcal{B}$, i.e., $X_{\mathcal{B}^r} = X_{\mathcal{B}}$, and for each layer $i \in I$, $|\mathcal{N}_i^r| \leq |\mathcal{N}_i|$. Then, CNF($\mathcal{B}^r$) $\subseteq$ CNF($\mathcal{B}$).*

5 Case Study

For our numerical evaluation, we apply our combinatorial cut-and-lift procedure to binary problems with normally distributed chance constraints. Recall that problem CC considers a linear objective and m constraints of the form

$$\mathbb{P}(\boldsymbol{a}_j^\top \boldsymbol{x} \leq b_j) \geq \epsilon_j, \qquad \forall j \in \{1, ..., m\}, \tag{7}$$

where $\boldsymbol{a}_j \in \mathbb{R}^n$ is a random variable vector with joint normal distribution $N(\boldsymbol{\mu}_j, \Sigma_j^2)$, μ_{ji} is the mean for each a_{ji}, and $\Sigma_j^2 \in \mathbb{R}^{n \times n}$ is the covariance matrix of $\boldsymbol{a}_j$. Values $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{\epsilon} \in [0.5, 1]^m$ are deterministic parameters. Each constraint (7) can be written as a non-linear inequality $\boldsymbol{\mu}_j^\top \boldsymbol{x} + \Phi^{-1}(\epsilon_j)||\Sigma_j \boldsymbol{x}||_2 \leq b_j$ for all $j \in \{1, ..., m\}$, where $\Phi(\cdot)$ is the cumulative distribution of a standard Gaussian and $||\cdot||_2$ is the Euclidean norm. In particular, this inequality is a special version of the second-order cone (SOC) constraint, i.e.,

$$\boldsymbol{\mu}^\top \boldsymbol{x} + \Omega \sqrt{\sum_{k \in \{1,...,l\}} (\boldsymbol{\sigma}_k^\top \boldsymbol{x} - d_k)^2} \leq b, \tag{8}$$

for a given vector $\boldsymbol{d} \in \mathbb{R}^l$, a constant $\Omega \in \mathbb{R}_+$, and matrix $\Sigma = [\boldsymbol{\sigma}_1, ..., \boldsymbol{\sigma}_l]^\top$ such that $\boldsymbol{\sigma}_k \in \mathbb{R}^n$ for all $k \in \{1, ..., l\}$.

We propose a novel BDD encoding for the general SOC inequalities (8) and evaluate its effectiveness for cases arising in normally distributed chance constraints. We also present a BDD encoding for chance constraints with independent distributions, i.e., where $\Sigma_j^2 \in \mathbb{R}^{n \times n}$ is a diagonal matrix. In this case, inequality (8) reduces to a SOC knapsack [3]:

$$\boldsymbol{\mu}^\top \boldsymbol{x} + \Omega \sqrt{\sum_{i \in I} \sigma_{ii}^2 x_i} \leq b. \tag{9}$$

6 Empirical Evaluation

Table 1. Aggregated results showing the overall performance of each technique for GCC.

	# Solve	Root Gap	Final Gap	Time (sec)	# Nodes	# Cuts	% Lifted
CPLEX	137	20.8%	7.7%	550.4	176,860.3	128	-
BW	139	20.0%	7.3%	409.5	252,865.5	31	-
BWL	150	15.7%	5.9%	280.8	150,200.1	23	90.7%
BG	166	13.5%	4.6%	210.0	84,592.0	324	-
BGL	**168**	**13.4%**	**4.4%**	**169.7**	**78,058.3**	132	72.0%

This section presents an empirical evaluation of our combinatorial cut-and-lift procedure for CC (see §5). We create a BDD for each of the m chance constraints and apply our procedure for each such constraint at the root node of the branch-and-bound tree. For any fractional point $\boldsymbol{x} \in [0, 1]^n$, we iterate over each BDD until one of them generates either a general or combinatorial BDD flow cut, as we describe in detail below. We then lift the inequality using Algorithm 4. The procedure ends when $\boldsymbol{x}$ cannot be cut-off.

We test our approach over the knapsack chance constraints (KCC) data set [3, 17] and generate a random set of instances for general chance constraints (GCC) (i.e., inequality (8) with $\boldsymbol{d} = \boldsymbol{0}$) following a similar procedure for

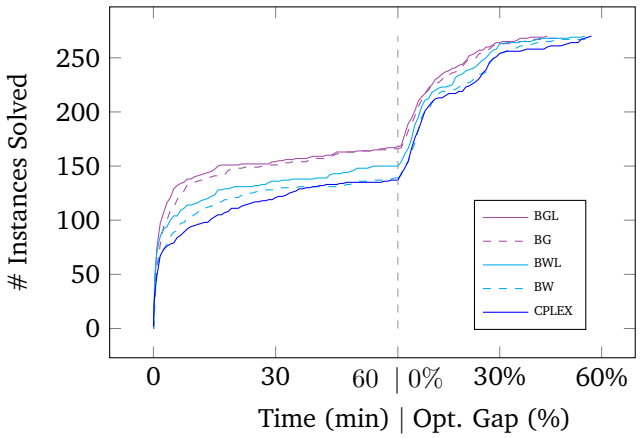


Figure 7. Profile plot comparing the accumulated number of instances solved over time (left), and the accumulated number of instances over a final gap range (right) for GCC dataset.

generating KCC. We consider $n \in \{75, 100, 125\}$, $m \in \{10, 20\}$, $\Omega \in \{1, 3, 5\}$, and density $2/\sqrt{n}$ over all the constraints. Parameters $\boldsymbol{\mu}_j$, Σ_j, and $\boldsymbol{c}$ are sampled from a discrete uniform distribution with $\boldsymbol{\mu}_j \in [-50, 50]^n$, $\Sigma_j \in [-20, 20]^{n \times n}$ and $\boldsymbol{c} \in [0, 100]^n$. Parameters b_j are given by

$$b_j = t \left(\sum_{k=1}^{n} \mu_{jk}^{+} + \Omega \sqrt{\sum_{i=1}^{n} \max \left\{ \sum_{k=1}^{n} \sigma_{jik}^{+}, \sum_{k=1}^{n} \sigma_{jik}^{-} \right\}^2} \right),$$

for all $j \in \{1, ..., m\}$ where $t \in \{0.1, 0.2, 0.3\}$ is the constraint tightness, $a^{+} := \max\{0, a\}$, and $a^{-} := \max\{0, -a\}$ for any $a \in \mathbb{R}$. Notice that b_j with $t = 0.3$ will remove approximately 50% of the possible assignments for $\boldsymbol{x} \in \{0, 1\}^n$. Then, we generate 5 random instances for each combination of n, m, Ω, and t, i.e., a total of 270 instances.

We implement four variants of our approach to test the BDD cuts and lifting procedure. The first two, BW and BWL, consider the weaker combinatorial BDD cuts (see §4.3), where BW omits the lifting procedure and BWL includes it. The other two variants, BG and BGL, use the combinatorial BDD flow cuts first and try the general BDD flow cuts (see §4.2) if the weaker approach fails to produce a cut. As before, BGL utilizes our lifting procedure in every generated constraint while BG does not. We use a BDD width limit (i.e., maximum number of nodes per layer) of 4000 nodes for all variants.

We also implement the cover cuts and lifting procedure for the KCC case [3]. We test their cover cuts with and without their continuous SOC lifting, C and CL, respectively, and also with our BDD lifting, BCL.

Our procedures are implemented in `C++` in the IBM ILOG CPLEX 12.9 solver using the `UserCuts` callback at the root node of the search. All experiments consider a single thread, a one-hour time limit, and the linearization strategy to solve the SOC problems.

6.1 Overall Performance

Table 1 presents the average results of all techniques for the GCC instances. The first column presents the number of problems solved to optimality. The second and third columns correspond to the average root gap and final gap across all instances. The fourth and fifth columns are the average solving time (including the BDD construction time) and nodes explored for the subset of instances that all techniques solve. The sixth column shows the number of cuts added by either the solver (i.e., for CPLEX) or our techniques. The last column presents the average percentage of original constraints that are lifted at least one time.

Table 1 shows that all our variants have better performance than CPLEX. BGL has the best performance solving 31 more instances than CPLEX. The difference on instances solved can be explained by the root node gap reduction for our approaches. Also, our lifting variants consistently solve more instances and are on average faster than BW and BG.

Figure 7 depicts the performance of each algorithm for the GCC instances. The graph illustrates the number of instances solved over time (left side) and the accumulated number of instances over a final gap range (right side). We see a clear dominance of our general BDD flow cuts (i.e., BG and BGL) and also the impact of lifting in number of instances solved and gap reduction. In particular, BG and BGL have the largest gap reductions and BWL has a consistently smaller gap than CPLEX. BW also improves upon CPLEX by a small margin.

7 Conclusions

We introduce a novel lifting and cutting-plane procedure for binary programs that leverage their combinatorial structure via a BDD encoding of their constraints. Our lifting procedure relies on 0-1 disjunctions to rotate valid inequalities and uses a BDD to efficiently compute the disjunctive sub-problems. While our combinatorial lifting can enhance any cutting-plane approach, we also propose a new BDD-based cut generation algorithm based on an alternative network-flow representation of the BDD.

BDDs give us the flexibility to apply our procedure to a wide range of non-linear problems. As a study case, we tested our procedure over normally distributed linear chance-constrained problems, a common application of SOC inequalities. To do so, we introduce a novel BDD encoding for these constraints and compare the performance of our procedure against a state-of-the-art solver and existing cut-and-lift procedure for SOC knapsacks. The empirical results show that our technique solves 31 more instances, reducing the final gap by 42.6% and having a threefold decrease in run-time over the general chance-constrained instances. In addition, our technique outperforms existing cut-and-lift methodologies for SOC knapsack problems by solving 17 more instances, achieving a 96.3% final gap reduction, and having comparable average solving time. We note that our procedure performs best when the solution set of each inequality is smaller and the quadratic term of the SOC inequalities is predominant.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., USA, 1993.

[2] Alper Atamtürk and Avinash Bhardwaj. Network design with probabilistic capacities. *Networks*, 71(1):16–30, 2018.

[3] Alper Atamtürk and Vishnu Narayanan. The submodular knapsack polytope. *Discrete Optimization*, 6(4):333–344, 2009.

[4] Egon Balas. Facets of the knapsack polytope. *Mathematical programming*, 8(1):146–164, 1975.

[5] Egon Balas. *Disjunctive Programming*. Springer, 2018.

[6] Markus Behle. Binary decision diagrams and integer programming. *Ph.D. Thesis*, 2007.

[7] David Bergman and Andre A Cire. Discrete nonlinear optimization by state-space decompositions. *Management Science*, 64(10):4700–4720, 2018.

[8] David Bergman, Andre A. Cire, Willem-Jan van Hoeve, and John N Hooker. Discrete optimization with decision diagrams. *INFORMS Journal on Computing*, 28(1):47–66, 2016.

[9] David Bergman and Leonardo Lozano. Decision diagram decomposition for quadratically constrained binary optimization. *Optimization Online e-prints*, 2018.

[10] Robert E Bixby, Mary Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling. Mixed-integer programming: A progress report. In *The sharpest cut: the impact of Manfred Padberg and his work*, pages 309–325. SIAM, 2004.

[11] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[12] Margarita P Castro, Andre A Cire, and J Christopher Beck. A combinatorial cut-and-lift procedure with an application to 0-1 second-order conic programming. *Under Review*, 2020.

[13] Abraham Charnes and William W Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

[14] Danial Davarnia and Willem-Jan van Hoeve. Outer approximation for integer nonlinear programs via decision diagrams. *Mathematical Programming*, Feb 2020.

[15] Peter L Hammer, Ellis L Johnson, and Uri N Peled. Facet of regular 0–1 polytopes. *Math. Prog.*, 8(1):179–206, 1975.

[16] IBM. *ILOG CPLEX Studio 12.9 Manual*, 2019.

[17] Seulgi Joung and Sungsoo Park. Lifting of probabilistic cover inequalities. *Operations Research Letters*, 45(5):513–518, 2017.

[18] Leonardo Lozano and J Cole Smith. A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. *Mathematical Programming*, pages 1–24, 2018.

[19] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. 1988.

[20] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. In *IPCO*, pages 348–360. Springer, 2001.

[21] Oleg V Shylo, Oleg A Prokopyev, and Andrew J Schaefer. Stochastic operating room scheduling for high-volume specialties under block booking. *INFORMS Journal on Computing*, 25(4):682–692, 2013.

[22] Christian Tjandraatmadja and Willem-Jan van Hoeve. Target cuts from relaxed decision diagrams. *INFORMS Journal on Computing*, 31(2):285–301, 2019.

[23] Cornelis Van de Panne and W Popp. Minimum-cost cattle feed under probabilistic protein constraints. *Management Science*, 9(3):405–430, 1963.

[24] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

Research Highlight: Sparse Regression at Scale: Branch-and-Bound rooted in First-Order Optimization

by HUSSEIN HAZIMEH, MIT OPERATIONS RESEARCH CENTER, *hazimeh@mit.edu*

This article is a summary of paper [12], which I co-authored with my advisor Rahul Mazumder and Ali Saab. We would like to thank the committee, which was chaired by Claudia d'Ambrosio and included Georgina Hall and Ruiwei Jiang, for the honorable mention. We would also like to thank Yongjia Song for the invitation to write this article.

1 Introduction

In linear regression, ℓ_0 regularization aims to find a small subset of features that leads to the best fit in terms of squared error. This foundational problem dates back to over five decades [13], with growing interest in the wider statistics, operations research, and computer science communities. This problem can lead to compact learning models that are easy to interpret, and thus it has important applications in critical domains such as healthcare and insurance.

In this work, we focus on the linear regression problem with a combination of ℓ_0 and ℓ_2 regularization. Given a data matrix $X \in \mathbb{R}^{n\times p}$ with n observations and p features, and a response vector $y \in \mathbb{R}^n$, the problem is defined as follows:

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0\|\beta\|_0 + \lambda_2\|\beta\|_2^2, \tag{1}$$

where the ℓ_0 (pseudo) norm, $\|\beta\|_0$, is defined as the number of nonzero entries in the vector β, and $\|\beta\|_2^2$ is the squared ℓ_2-norm of β (also referred to as ridge regularization). The regularization parameter $\lambda_0 \geq 0$ controls the number of nonzeros (selected variables), and $\lambda_2 \geq 0$ controls the amount of shrinkage imposed by the ℓ_2 regularizer. Problem (10) with $\lambda_2 = 0$ leads to the classical best subset selection problem. If $\lambda_2 > 0$, then the ridge regularizer can mitigate overfitting in certain settings—see [10, 15, 11] for further discussions on this matter. Recently there has been exciting work on developing Mixed Integer Programming (MIP)-based approaches to solve (10), e.g., [3, 17, 6, 4, 11, 19]. Current work shows that under many important high-dimensional regimes, ℓ_0-based estimators possess statistical properties (variable selection, prediction, and estimation) that are superior to computationally friendlier alternatives such as ℓ_1 regularization (Lasso) [18] and stepwise regression—see [11] for an in-depth discussion.

Despite its appeal, Problem (10) is NP-Hard [16] and poses computational challenges. [3] demonstrated that modern MIP solvers can handle instances with p up to a thousand. Larger instances can be handled when λ_2 is sufficiently large and the feature correlations are low, e.g., see [4]; and also [5] for the classification variant of Problem (10). While these state-of-the-art global optimization approaches show promising results, they are still relatively slow for practical usage [10, 11], as they typically take in the order of minutes to days to obtain optimal solutions. For example, our experiments show that these methods cannot terminate in two hours for typical instances with $p \sim 10^4$. On the other hand, the fast Lasso solvers, e.g., `glmnet` [8], and approximate methods for (10), such as `L0Learn` [11], can handle much larger instances, and they typically terminate in the order of milliseconds to seconds.

Our goal in this work is to speed up the global optimization of Problem (10). In particular, we aim to (i) reduce the run time for solving typical instances with $p \sim 10^4$ from hours to seconds, and (ii) scale to larger instances with $p \sim 10^7$ in reasonable times (order of seconds to hours). Scaling to such instances will facilitate the adoption of ℓ_0 regularization in practice and allow for a better understanding of its empirical properties. To this end, we propose a specialized, nonlinear branch-and-bound (BnB) framework for solving Problem (10) to certified optimality.

One of the distinguishing features of our framework is a specialized first-order optimization method for solving the node subproblems in the BnB tree. The first-order method heavily exploits the problem structure to speed up computation and reduce memory requirements. Our framework is open-source and freely available at https://github.com/alisaab/l0bnb. This makes our approach different from prior work on global optimization for Problem (10), which rely on commercial MIP solvers such as Gurobi and MOSEK. These MIP solvers are also based on a BnB framework, but they are equipped with general-purpose subproblem solvers and heuristics that do not take into account the specific structure in Problem (10). In the next section, we give an overview of our approach.

2 Overview of Our Approach

2.1 MIP Formulation

Two MIP formulations for Problem (10) have been commonly used in recent literature. The first uses Big-M constraints to model the ℓ_0 norm. The second uses a perspective reformulation [7, 9, 6] and does not require a Big-M. In [12], we discuss how each of these two formulations has unique advantages. For example, each formulation has a range of λ_0 and λ_2 for which it leads to tighter relaxations than the other formulation. To combine the advantages of these two formulations, we merge them into one hybrid formulation. Next, we introduce the hybrid formulation. We assume that there is a finite scalar M (a-priori specified) such that an optimal solution of (10), say β^*, satisfies: $\|\beta^*\|_\infty \leq M$. Then the hybrid formulation is defined as follows:

$$\min_{\beta,z,s} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_0 \sum_{i=1}^{p} z_i + \lambda_2 \sum_{i=1}^{p} s_i \tag{2a}$$

$$\text{s.t.} \quad \beta_i^2 \leq s_i z_i, \qquad i \in \{1, 2, \ldots, p\} \tag{2b}$$

$$-M z_i \leq \beta_i \leq M z_i, \; i \in \{1, 2, \ldots, p\} \tag{2c}$$

$$z_i \in \{0, 1\}, s_i \geq 0, \quad i \in \{1, 2, \ldots, p\} \tag{2d}$$

The perspective constraints (11b) and the Big-M constraints (11c) enforce: $z_i = 0 \implies \beta_i = 0$. Thus, the term $\sum_{i=1}^{p} z_i$ represents the ℓ_0 norm of β. Moreover, each continuous variable s_i represents β_i^2 (in particular, at any optimal solution (β^*, z^*, s^*), we have $s_i^* = (\beta_i^*)^2$ for all i). These observations can be used to show that formulation (11) is equivalent to Problem (10). Note that formulation (11) can be represented as a mixed integer second order cone program (MISOCP) and can be handled using existing MISOCP solvers such as Gurobi and MOSEK. However, these solvers face difficulties in scaling beyond $p \sim 10^3$. Next, we present a specialized BnB framework for solving Problem (11) at larger scales.

2.2 A Specialized Branch-and-Bound (BnB) Framework

In this section, we give a high-level overview of our specialized nonlinear BnB framework for solving formulation (11). First, we briefly recall the high-level mechanism behind nonlinear BnB, in the context of our problem.

Nonlinear BnB at a Glance: The algorithm starts by solving a nonlinear relaxation for (11), i.e., the root node. Then, it selects a branching variable, say variable $j \in \{1, 2, \ldots, p\}$, and creates two new nodes (optimization subproblems): one node with $z_j = 0$ and another with $z_j = 1$, where all the other z_i's are relaxed to the interval $[0, 1]$. For every unvisited node, the algorithm proceeds recursively, i.e., by solving an optimization subproblem at the current node and then branching on a new variable to create two new nodes. This leads to a search tree with nodes corresponding to optimization subproblems and edges representing branching decisions.

To reduce the size of the search tree, our BnB prunes a node (i.e., does not branch on it) in either one of the following situations: (i) the subproblem at the current node has an integral z or (ii) the objective of the current subproblem exceeds the best available upper bound on (11). The subproblem need not be solved exactly; lower bounds (a.k.a. dual bounds) can be used instead.

Our discussion above outlines how nonlinear BnB operates in general. Of course, the specific strategies used, such as solving the relaxations, passing information across the nodes, and selecting branching variables, can have a key impact on scalability. In what follows, we give an overview of our strategies.

A Primal Relaxation Solver: Unlike the state-of-the-art approaches for nonlinear BnB, which employ primal-dual relaxation solvers [2], we rely solely on a primal method. Specifically, we design a highly scalable coordinate descent (CD)-based algorithm for solving the continuous node subproblems corresponding to formulation (11). Our CD

operates on subproblems that are reformulated in the β space (as opposed to the extended (β, s, z) space) where the Big-M and perspective constraints are eliminated. The algorithm heavily shares and exploits warm starts, active sets, and information on the gradients, across the BnB tree. In contrast, interior point methods, which are commonly used in commercial non-linear BnB solvers, face difficulties in exploiting sparsity and warm starts.

Dual Bounds: Dual bounds on the subproblems are required by the BnB for search space pruning, yet our relaxation solver works in the primal space for scalability considerations. Thus, we develop a new efficient method for obtaining dual bounds directly from (approximate) primal solutions. A key observation we exploit in this method is that by fixing some dual variables, the dual problem can be partially maximized in closed form. We provide an analysis of this method and show that the tightness of the corresponding dual bounds depends on the sparsity level and *not* on the number of features p; this serves as a theoretical justification for why our proposed method works well in high dimensions.

Branching and Incumbents: We develop an efficient variant of strong branching [1], which leverages the solutions and active sets of previous node relaxations to make optimization tractable. Moreover, we employ several efficient heuristics to obtain good incumbents.

The mathematical formulations and details of the strategies above are discussed in [12].

3 Experiments

We perform a series of high-dimensional experiments to study the run time of our BnB and compare to state-of-the-art approaches. While our dataset and parameter choices are well-grounded from a statistical perspective, we note that our goal here is not to study the statistical properties of ℓ_0 estimators. We refer the reader to [3, 4, 11] for empirical studies of the statistical properties.

3.1 Experimental Setup

Synthetic Data Generation: We generate a multivariate Gaussian data matrix with samples drawn from MVN$(0, \Sigma_{p\times p})$, a sparse coefficient vector $\beta^\dagger \in \mathbb{R}^p$ with $k^\dagger$ equi-spaced nonzero entries all set to 1, and a noise vector $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. The response is then obtained from the linear model $y = X\beta^\dagger + \epsilon$. We define the *signal-to-noise ratio (SNR)* as follows: SNR$=$ Var$(X\beta^\dagger)/\sigma^2$. In all the experiments, we set σ^2 to achieve SNR$=5$—this is a relatively difficult setting which still allows for full support recovery, under suitable choices of n, p, and Σ (see [11] for a discussion on appropriate levels of SNR).

Warm Starts, λ_0, λ_2, and M: We obtain the warm start from `L0Learn`[3] [11] and use it for all the MIP solvers considered. Unless otherwise specified, we fix λ_0 to a value λ_0^* which leads to a support size of $k^\dagger$ (i.e., that of the true model). The parameters λ_2 and M can affect the run time significantly, so we study the sensitivity to these choices in our experiments. We consider choices of λ_2 that are relevant from a statistical perspective. We define λ_2^* as the λ_2 which minimizes the ℓ_2 estimation error, among all $k^\dagger$-sparse solutions. More formally, for every λ_0, λ_2, let $\beta(\lambda_0, \lambda_2)$ denote an optimal solution to Problem (10). We define λ_0^* and λ_2^* as a solution of:

$$\lambda_0^*, \lambda_2^* \in arg\,min_{\lambda_0,\lambda_2\geq 0} \|\beta^\dagger - \beta(\lambda_0, \lambda_2)\|_2$$
$$\text{s.t. } \|\beta(\lambda_0, \lambda_2)\|_0 = k^\dagger.$$

We estimate λ_0^* and λ_2^* using `L0Learn` by doing a two-dimensional grid search over λ_0 and λ_2; in particular, $\lambda_2 \in [10^{-4}, 10]$ and the range for λ_0 is selected automatically by the toolkit. In the experiments, we report our λ_2 choices as a fraction or multiple of λ_2^* (e.g., $\lambda_2 = 0.1\lambda_2^*$). Moreover, we define M^* as the minimum M for which the hybrid formulation (11) is equivalent to (10). We estimate M^* as the ℓ_∞ norm of the solution obtained by `L0Learn`, and we report our choices in terms of M^*. Note that in almost all cases considered, the support of $\beta^\dagger$ is correctly recovered by `L0Learn`, which yields high quality estimates for M^* and λ_2^*.

Solvers and Settings: Our solver, L0BnB[4], is written in Python with critical code sections optimized using Numba [14]. We compare L0BnB with Gurobi and MOSEK on formulation (11). We also compare against [4] who solve the cardinality-constrained variant of (10); here we set the number of nonzeros to $k^\dagger$. For all approaches, we set the threshold for the relative optimality gap[5] to 1.0%.

[3] We use the default CD-based algorithm in `L0Learn`. We remind the reader that `L0Learn` is a local optimization method that does not provide certificates of global optimality.

[4] https://github.com/alisaab/l0bnb

[5] Given the upper bound UB and lower bound LB, the relative optimality gap is defined as (UB − LB)/UB.

3.2 Comparison with the State of the Art

In this section, we study the scalability of the different solvers in terms of the number of features p. We generate synthetic datasets with $n = 10^3$, $p \in \{10^3, 10^4, 10^5, 10^6\}$, and $k^\dagger = 10$. We consider a constant correlation setting, where $\Sigma_{ij} = 0.1\ \forall i \neq j$ and 1 otherwise. The parameters λ_2 and M can have a significant effect on the run time. Thus, we report the timings for different choices of these parameters. In particular, in Table 2 (top panel), we fix $M = 1.5M^*$ and report the timings for the choices $\lambda_2 \in \{0.1\lambda^*, \lambda^*, 10\lambda^*\}$ (where λ_2^* and M^* are defined in Section 3.1). In Table 2 (bottom panel), we fix $\lambda_2 = \lambda_2^*$ and report the timings for $M \in \{M^*, 2M^*, 4M^*, \infty\}$.

Table 2. (Sensitivity to λ_2 and M) Running time in seconds for solving (10) (via formulation (11)) by our proposal (L0BnB), Gurobi (GRB) and MOSEK (MSK). The method [4] solves the cardinality-constrained variant of (10). The symbols "*" or "-" mean that the method does not terminate in 2 hours. "**" means that the method terminates before 2 hours due to insufficient memory (30GB). Optimality gap is shown in parentheses for "*" and "**", and is 100% for "-". Choices of λ_2 and M are discussed in the text.

	$\lambda_2 = \lambda_2^*$				$\lambda_2 = 0.1\lambda_2^*$				$\lambda_2 = 10\lambda_2^*$			
p	L0BnB	GRB	MSK	[4]	L0BnB	GRB	MSK	[4]	L0BnB	GRB	MSK	[4]
10^3	**6**	95	216	1079	**2**	81	273	-	**0.01**	2372	54	0.4
10^4	**14**	-	5354	*(45%)	**33**	-	6693	-	0.8	-	1511	**0.6**
10^5	**255**	-	*(35%)	*(70%)	**544**	-	*(41%)	-	**10**	-	*(3%)	13
10^6	**3468**	-	-	*(88%)	***(7%)**	-	-	-	**43**	-	-	4123

	$M = M^*$			$M = 2M^*$			$M = 4M^*$			$M = \infty$		
p	L0BnB	GRB	MSK	L0BnB	GRB	MSK	L0BnB	GRB	MSK	L0BnB	GRB	MSK
10^3	**0.7**	35	106	**1**	199	279	**1**	1636	259	**1**	2307	336
10^4	**2**	-	1909	**21**	-	6646	**23**	-	*(7%)	**23**	-	-
10^5	**25**	-	*(9%)	**543**	-	*(59%)	**588**	-	-	**628**	-	-
10^6	**309**	-	-	**7180**	-	-	****(3%)**	-	-	****(3%)**	-	-

The results in the top panel of Table 2 indicate significant speed-ups, reaching over $200,000$x compared to Gurobi, 5000x compared to MOSEK, and 3600x compared to [4]. At $\lambda_2 = \lambda_2^*$, L0BnB is the only solver that can handle $p \geq 10^5$, and can, in fact, handle $p = 10^6$ in less than an hour. Recall that λ_2^* minimizes the ℓ_2 estimation error and leads to an estimator that is the closest to the ground truth.

For $\lambda_2 = 0.1\lambda_2^*$, L0BnB is again the only solver that can handle $p \geq 10^5$. For $\lambda_2 = 10\lambda_2^*$, the optimization problem seems to become easier: L0BnB is up to 600x faster compared to λ_2^*, and [4] can handle up to 10^6. However, L0BnB in this case, is ~ 100 times faster than [4] at $p = 10^6$. The speed-ups for $\lambda_2 = 10\lambda_2^*$ can be attributed to the fact that a larger λ_2 adds a large amount of strong convexity to the objective (via the perspective term)—improving the performance of the relaxation solvers[6]. It is worth emphasizing that our L0BnB is prototyped in Python; as opposed to the highly efficient BnB routines available in commercial solvers such as Gurobi and MOSEK.

Ideally, we desire a solver that can address (10) over a range of λ_2 values, which includes values in the neighborhood of λ_2^*. However, the results in Table 2 suggest that the state-of-the-art methods (except L0BnB) seem to only work for quite large values of λ_2 (which in this case, do not correspond to solutions that are interesting from a statistical viewpoint). On the other hand, L0BnB seems to be the only method that can scale to $p \sim 10^6$ while being relatively robust to the choice of λ_2.

In the bottom panel of Table 2, the results also indicate that L0BnB significantly outperforms Gurobi and MOSEK for different choices of M. For all the solvers, the run time increases with M, and the longest run times are for $M = \infty$, which corresponds to the (pure) perspective formulation. However, even with $M = \infty$, L0BnB can still achieve a decent gap (3%) for $p = 10^6$ in less than two hours.

Additional ablation studies and experiments on real data are presented in [12].

[6] Gurobi is the only exception to this observation. We investigated this: Gurobi generates additional cuts only for the case of $10\lambda_2^*$, which seems to slow down the relaxation solver.

4 Conclusion

We considered the exact computation of estimators from the least squares problem regularized with a combination of the ℓ_0 and ℓ_2 norms. We developed a highly specialized nonlinear BnB framework for solving the problem. To solve the node subproblems in BnB, we designed a scalable first-order method, unlike state-of-the-art MIP solvers which rely on primal-dual methods. Our first-order method consists of coordinate descent along with active set updates and gradient screening, which exploit the information shared across the search tree to reduce the coordinate update complexity. Moreover, we proposed a new method for obtaining dual bounds from the primal coordinate descent solutions and showed that the quality of these bounds depends on the sparsity level, rather than the number of features. Experiments on both real and synthetic datasets indicate that our method is over 3600x faster than existing solvers, handling high-dimensional instances with $p = 8.3 \times 10^6$ in the order of seconds to a few minutes. Our work demonstrates that carefully designed first-order methods can be highly effective within a BnB framework; and can perhaps, be applied to more general mixed integer programs involving sparsity.

References

[1] Applegate, D., Bixby, R., Cook, W., Chvátal, V.: On the solution of traveling salesman problems (1998)

[2] Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-integer nonlinear optimization. Acta Numerica **22**, 1–131 (2013)

[3] Bertsimas, D., King, A., Mazumder, R., et al.: Best subset selection via a modern optimization lens. The Annals of Statistics **44**(2), 813–852 (2016)

[4] Bertsimas, D., Van Parys, B.: Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. arXiv preprint arXiv:1709.10029 (2017)

[5] Dedieu, A., Hazimeh, H., Mazumder, R.: Learning sparse classifiers: Continuous and mixed integer optimization perspectives. arXiv preprint arXiv:2001.06471 (2020)

[6] Dong, H., Chen, K., Linderoth, J.: Regularization vs. Relaxation: A conic optimization perspective of statistical variable selection. ArXiv e-prints (2015)

[7] Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0–1 mixed integer programs. Mathematical Programming **106**(2), 225–236 (2006)

[8] Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software **33**(1), 1–22 (2010). URL http://www.jstatsoft.org/v33/i01/

[9] Günlük, O., Linderoth, J.: Perspective reformulations of mixed integer nonlinear programs with indicator variables. Mathematical programming **124**(1-2), 183–205 (2010)

[10] Hastie, T., Tibshirani, R., Tibshirani, R.J.: Extended comparisons of best subset selection, forward stepwise selection, and the lasso. arXiv preprint arXiv:1707.08692 (2017)

[11] Hazimeh, H., Mazumder, R.: Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. ArXiv e-prints (2018)

[12] Hazimeh, H., Mazumder, R., Saab, A.: Sparse regression at scale: Branch-and-bound rooted in first-order optimization. arXiv preprint arXiv:2004.06152 (2020)

[13] Hocking, R.R., Leslie, R.: Selection of the best subset in regression analysis. Technometrics **9**(4), 531–540 (1967)

[14] Lam, S.K., Pitrou, A., Seibert, S.: Numba: A llvm-based python jit compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, pp. 1–6 (2015)

[15] Mazumder, R., Radchenko, P., Dedieu, A.: Subset selection with shrinkage: Sparse linear modeling when the snr is low. arXiv preprint arXiv:1708.03288 (2017)

[16] Natarajan, B.K.: Sparse approximate solutions to linear systems. SIAM journal on computing **24**(2), 227–234 (1995)

[17] Pilanci, M., Wainwright, M.J., El Ghaoui, L.: Sparse learning via boolean relaxations. Mathematical Programming **151**(1), 63–87 (2015)

[18] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996)

[19] Xie, W., Deng, X.: Scalable algorithms for the sparse ridge regression (2020)