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

J. Cole Smith, PhD
Associate Provost for Academic Initiatives
Professor of Industrial Engineering
Clemson University, jcsmith@clemson.edu

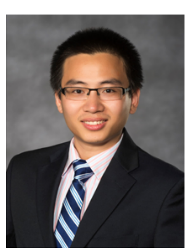
My term as Chair could hardly have come at a more exciting time for the INFORMS Computing Society (ICS). Concepts regarding data and optimization that were formerly esoteric or niche are now discussed in common conversation. Many laypeople now have a broad (if sometimes imprecise) understanding of concepts like smart grid optimization, intelligent pricing, healthcare optimization, optimization in manufacturing, and telecommunication and transportation system design. ICS members have long been at the forefront of these areas. As complementary algorithmic schemes and computational hardware and software systems evolve, ICS members will have a terrific opportunity to impact the field of optimization, analysis, and data-driven decision making.

Yet, our field must prepare for several challenges that confront our field. We must diversify our society to become the inclusive group that we wish to be. We need to demonstrate to the outside world what kind of impact our research and education makes in modeling and solving real-world problems. Alongside this message, we must emphasize the importance of the foundational and theoretical advances that our community makes, which enable the next generation of algorithms. Finally, the ICS needs to continue to proactively seek ways of supporting our researchers and students. I am looking forward to seeing how we can enhance our community's success in these areas in 2019.

I am grateful for the enthusiasm, experience, and leadership of those who have been so active in the ICS. Enjoy the newsletter, and we look forward to seeing you in the near future at INFORMS conferences!

## Message from the Editor

Yongjia Song, PhD
Assistant Professor, 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, a memorial article for Harvey J. Greenberg, Founding Editor of the *INFORMS Journal on Computing*, as well as research highlights and insights for the 2018 ICS awarding papers. Special thanks to all who contribute to this newsletter! This is the very first newsletter that I have edited since I took over this role. I would like to express my gratitude to former ICS newsletter editors, Dr. Yongpei Guan and Dr. Jeff Linderoth for their help. I will also greatly appreciate any comment or suggestion that you may have for the newsletter.

# 2018 ICS Prize

**The 2018 ICS Prize goes to James V. Burke (University of Washington), Frank E. Curtis (Lehigh University), Adrian S. Lewis (Cornell University), and Michael L. Overton (New York University)** for their pioneering work on gradient sampling methods for nonsmooth optimization, as detailed in the papers: (1) "Approximating Subdifferentials by Random Sampling of Gradients." Mathematics of Operations Research, 27:567-584, 2002; (2) "A Robust Gradient Sampling Algorithm for Nonsmooth, Nonconvex Optimization." SIAM J. Optimization, 15(3):751-779, 2005; (3) "A Sequential Quadratic Programming Algorithm for Nonconvex, Nonsmooth Constrained Optimization." SIAM J. Optimization, 22(2):474-500, 2012; (4) "A BFGS-SQP method for Nonsmooth, Nonconvex, Constrained Optimization and its Evaluation using Relative Minimization Profiles." Optimization Methods and Software, 32(1):148-181, 2017; (5) "Gradient Sampling Methods for Nonsmooth Optimization." arXiv:1804.11003.

Committee members: Andreas Wächter (Northwestern University, Chair), Fatma Kilinc-Karzan (Carnegie Mellon University) and Jean-Paul Watson (Sandia National Laboratory).

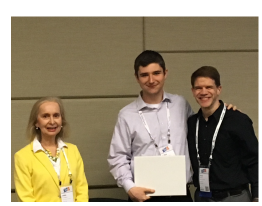
# The 2018 ICS Best Student Paper Award

**Winner: Aleksandr M. Kazachkov, Carnegie Mellon University (now a postdoc at Polytechnique Montreal).** Award-winning paper: "V-Polyhedral Disjunctive Cuts" This paper develops a novel approach - based on so-called V-Polyhedral Cuts (VPCs) - for generating valid inequalities when solving mixed-integer linear programming (MILP) problems. The cuts are motivated by several shortcomings of existing cut generation techniques, such as issues related to numerical instability and a "tailing off" effect when they are used recursively. The use of VPCs mitigates such effects by providing a practical method for generating strong cuts without recursion. Theoretical properties of such cuts are presented and computational tests of their performance are conducted. The computational results indicate that the cuts generated are strong and that there appear to exist classes of MILP instances for which VPCs work especially well.

**Honorable mentions:**

**Colin P. Gillen, University of Pittsburgh**, "Fortification Against Cascade Propagation Under Uncertainty"

**Chris Lourenco, Texas A&M University**, "Asymptotically Optimal Exact Solution of Sparse Linear Systems via Left-Looking Roundoff-Error-Free LU Factorization"



Committee members: Sergiy Butenko (Texas A&M University), Frank E. Curtis (Lehigh University) and Anna Nagurney (University of Massachusetts Amherst, Chair).

# A Tribute to Harvey J. Greenberg

by Allen Holder[1]

The INFORMS Computing Society lost one its most effectual patrons, Harvey J. Greenberg, on June 29, 2018. Harvey cofounded our society in the pre-INFORMS era of the Operations Research Society of America (ORSA), at which time our society was called the Computer Science Special Interest Group. He subsequently served as our second chair and started the ORSA Journal on Computing, now known as the INFORMS Journal on Computing. He continued to promote and serve our society throughout his career by organizing conferences and symposia, by spearheading online education, and by encouraging us to adapt to, prepare for, and embrace novel and emerging research. The ICS initiated the Harvey J. Greenberg Award for Service in 2007 in recognition of his lifelong efforts to advance and serve our society. Harvey further impacted many of our lives through his research, friendship, and encouragement.

Two memorials already note many of Harvey's professional accomplishments, and I encourage those who have not already read these to do so.

- A Memorial to Harvey J. Greenberg, Founding Editor of the *INFORMS Journal on Computing*, by A. Holder, F. Murphy, and W. Pierskalla, doi=10.1287/ijoc.2018.0843
- In Memoriam: Harvey J. Greenberg, 1940-2018, by Peter Horner, *ORMS Today*, vol. 45, num. 3, June, 2018

I want to give a more personal perspective here, as Harvey's relationship with our society was more intimate compared with the broad intentions of these earlier homages. That said, I have struggled to author a succinct review that would adequately express what Harvey meant to me, and I suspect to many of us. The idea of including my eulogy came to mind as I contemplated yet another attack, and I hope this will suffice. Below is a slightly edited version of my spoken comments at Harvey's Memorial last July.

*Students don't enter graduate school with any real hope of fostering a family-like relationship with their advisors. No, most simply long to survive and to satisfy their advisors' standards. There are, of course, professional expectations if all goes well, for instance reference letters for an initial job search, but the affectionate and familial embrace of your advisor as a colleague are most often wistful daydreams. But sometimes daydreams come true. My relationship with Harvey began cordially and professionally as would any student-teacher interaction, but it grew into a warm and amiable kinship, largely due to Harvey's ability to see in me what I could not. He nurtured my academic growth by giving me confidence when I needed it, by giving me resources and space to pursue my interests, and by challenging me with what sometimes seemed to be insurmountable tasks. Our relationship was that of fast friends by graduation, so much so that he didn't flinch the day when he caught me fixing his computer after breaking into his house. He was curious about how I had gained entry, but the joy of a working computer far outweighed any concern. Our friendship continued to advanced after graduation, and our closeness matured into one of those rare and cherished lifelong bonds between old friends. Academics are fond to relay the moniker of Academic Father/Mother's on their advisors, but in my case, our relationship blossomed heavily on the fraternal intent of this title. Harvey's death weighs on me as though I have lost a member of my inner family.*

*I have been advised to keep my comments brief, restricting all that Harvey was to me to a mere few moments. I want to use this short time to commemorate Harvey as the person that I knew him to be. Indeed, Harvey's professional accolades and contributions were the natural byproducts of Harvey being, well, Harvey, and by remembering him, maybe we can all find ideals that can inspire us. Here are a few of the characteristics that help portray Harvey as I knew and loved him.*

### Harvey was all in

*Harvey's commitment was absolute, an intimidating attribute that often made me question my own dedications. His pursuits were not lukewarm ambitions that could be chased as time permitted, but rather, they were the passions of life that deserved focus, care, and dedication. Harvey knew no sense of hesitancy, and his quick decisiveness could be jarring. I witnessed this characteristic professionally, e.g. his near instantaneous flip to computational biology, and I heard of it personally, e.g. his immediate love for Ellie. The story of the latter as I recall was that Harvey and Ellie met while hiking, and Harvey was so immediately taken by Ellie that he called to negotiate for her hand within a few days of his return home. I remember hearing this story for the first time and thinking, wow, that is so "Harvey."*

### Harvey was nonstop

*I don't know what Harvey would have called "idle." He certainly knew that this word was in the common lexicon, but it must have seemed foreign to him. Tasks simply couldn't get done fast enough, and all moments seemed to be packed with the possibility of accomplishment. Working with Harvey was a sort of cognitive athletics, and it seemed like he was always pushing my intellectual fitness. His quick mind could create, consider, and accept or reject thoughts much, much faster than I, and I constantly felt like I needed to stop, catch my breath, and contemplate. But Harvey would just move*

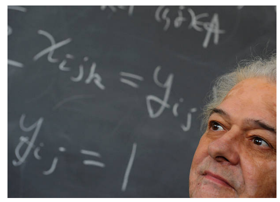
[1]Department of Mathematics, Rose-Hulman Institute of Technology. holder@rose-hulman.edu

*on, and on, and on, even as I tried to pause and register the moment. I was most keenly aware of Harvey's ceaselessness twice during my studies, the first of which followed my Ph.D. exams. I was simply exhausted from months of studying (my days had often started at 4:00 or 5:00 in the morning, which was probably about the time that Harvey had been going to bed), but I passed. Harvey offered to treat me to lunch immediately after my oral exam, and I thought great, a bit of celebration and then a short rest. But Harvey had had other plans, and his legal pad hit the table before our food. He was happy that I had passed, but that had already happened, and well, there were new problems to tackle. The other moment was when I submitted my first draft of my thesis, and though I knew Harvey well at that point, I had somehow erroneously assumed that I would get a week or two before he completed his review. I arrived at my office the very next morning to find light peeking from under Harvey's door, and I cringed. He returned the draft a few moments later in a sea of red marks, and with a wry smile he chuckled and went home to catch some sleep.*

**Harvey had a youthful buoyancy**

*Harvey could certainly be focused, but he could also be lighthearted. He had a giggle about him, a giggle that often followed a play on words or a twist in meaning. Such antics occurred with rapidity as he taught, as he communicated with friends and colleagues, and as he sat and pondered with himself. I once asked during a class about the definition of "vexing," a term that he had used so often that I thought it must have a specific mathematical meaning. He smiled and giggled as he replied that it was just a play on the mathematical term "convex," and that he employed the pun to impugn the lack of this property. I mean really, how was I supposed to know that an optimization problem could be giddy with distress? It was as if a silliness accompanied his thoughts, a silliness that was innocent and jovial and that helped him progress. Harvey's all-in and nonstop manner could awkwardly blind him to his surroundings and could lead to hilarious moments. There are several such stories in the memoir we created at his retirement, but my favorite occurred during our trip to the Netherlands. We had traveled to Amsterdam, and Harvey and I were engrossed in our mathematical conversation when he looked up and proclaimed, "Ah, the Dutch, they are so festive." I almost couldn't contain my laughter as I alerted Harvey that we had wandered into the red light district - he stopped, looked around and said "oh that explains it," and then it was right back to math.*

**Harvey was keenly insightful**



*Although Harvey was regularly aloof to the things which he chose not to pursue, he was perceptively caring about the things that he chose to pursue. Those who were close to him could depend on his assistance, even if they didn't seek or want it. This could certainly be a proverbial helping-hand with a current problem, but what made Harvey special was his ability to see ahead. He nurtured and willed many of our professional stalwarts into existence because he foresaw their benefits, and he would often nudge me toward opportunities that he knew would help me. I think our profession will miss most his vision, and I will certainly miss his guidance.*

**Harvey had an altruistic heart**

*Harvey had high ideals, and he was quick to voice them. He had a deep conviction for the greater good, and he was willing to risk for it. He really wanted his life's work to count, to make a difference, and to help others, and he felt spurned when hollow rules of merit seemed to undervalue his efforts. I want to end by saying that Harvey accomplished his big goal, and that very many indeed have benefited from his life's work, of which I'm only a minor representative. I so wish him comfort in this knowledge, and I thank him for everyone he touched.*

# Research Highlight: The Gradient Sampling Methodology

by James V. Burke[2], Frank E. Curtis[3], Adrian S. Lewis[4] and Michael L. Overton[5]

## 1 Introduction

The principal methodology for minimizing a smooth function is the steepest descent (gradient) method. One way to extend this methodology to the minimization of a nonsmooth function involves approximating subdifferentials through the random sampling of gradients. This approach, known as *gradient sampling* (GS), gained a solid theo-

[2]Department of Mathematics, University of Washington, Seattle, WA. jvburke01@gmail.com
[3]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA. frank.e.curtis@gmail.com
[4]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY. adrian.lewis@cornell.edu
[5]Courant Institute of Mathematical Sciences, New York University. mo1@nyu.edu

retical foundation about a decade ago [BLO05, Kiw07], and has developed into a comprehensive methodology for handling nonsmooth, potentially nonconvex functions in the context of optimization algorithms. In this article, we summarize the foundations of the GS methodology, provide an overview of the enhancements and extensions to it that have developed over the past decade, and highlight some interesting open questions related to GS.

## 2 Fundamental Idea

The central idea of gradient sampling can be explained as follows. When a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at a point $x$ (at which $\nabla f(x) \neq 0$), the traditional steepest descent direction for $f$ at $x$ in the 2-norm is found by observing that

$$\arg \min_{\|d\|_2 \leq 1} \nabla f(x)^T d = -\frac{\nabla f(x)}{\|\nabla f(x)\|_2}; \tag{1}$$

in particular, this leads to calling the negative gradient, namely, $-\nabla f(x)$, the direction of steepest descent for $f$ at $x$. However, when $f$ is not differentiable near $x$, one finds that following the negative gradient direction might offer only a small amount of decrease in $f$; indeed, obtaining decrease from $x$ along $-\nabla f(x)$ may be possible only with a very small stepsize. The GS methodology is based on the idea of stabilizing this definition of steepest descent by instead finding a direction to *approximately* solve

$$\min_{\|d\|_2 \leq 1} \max_{g \in \bar{\partial}_\epsilon f(x)} g^T d, \tag{2}$$

where $\bar{\partial}_\epsilon f(x)$ is the $\epsilon$-subdifferential of $f$ at $x$ [Gol77]. To understand the context of this idea, recall that the (Clarke) subdifferential of a locally Lipschitz $f$ at $x$, denoted $\bar{\partial} f(x)$, is the convex hull of the limits of all sequences of gradients evaluated at sequences of points, at which $f$ is differentiable, that converge to $x$ [Cla75]. The $\epsilon$-subdifferential, in turn, is the convex hull of all subdifferentials at points within an $\epsilon$-neighborhood of $x$. Although the $\epsilon$-subdifferential of $f$ at $x$ is not readily computed, the central idea of gradient sampling is to approximate the solution of (2) by finding the smallest norm vector in the convex hull of gradients computed at randomly generated points in an $\epsilon$-neighborhood of $x$, then normalizing the result to have unit norm. See [BLO02] for analysis on approximating an $\epsilon$-subdifferential by sampling gradients at randomly generated points.

A complete algorithm based on the GS methodology is stated as Algorithm 1, taken from the recent survey paper [BCL$^+$19]. To illustrate the efficacy of this algorithm compared to more standard gradient and subgradient methodologies, let us show its performance on a nonsmooth variant of the nonconvex Rosenbrock function [Ros60], namely,

$$f(x) = 8|x_1^2 - x_2| + (1 - x_1)^2. \tag{3}$$

The contours of this function are shown in Figure 1; the black asterisk indicates the initial iterate $x^0 = (0.1, 0.1)$ and the red asterisk indicates the unique minimizer $x^* = (1, 1)$. The blue dots show the iterates generated by the gradient sampling method (Algorithm 1) converging to $x^*$, roughly tracing out the parabola on which $f$ is nonsmooth, but never actually landing on it, even to finite precision. In contrast, the magenta dots show the iterates of the gradient method with the same line search enforcing (4) from Algorithm 1, indicating that these iterates move directly toward the parabola on which $f$ is nonsmooth and stall without moving along it toward the minimizer $x^*$. The essential difficulty is that the direction of descent tangential to the parabola is overwhelmed by the steepness of the graph of the function near the parabola. The gradient sampling method, by choosing the direction of least norm in the convex hull of sampled gradients, is able to approximate the tangential directions of descent toward $x^*$.

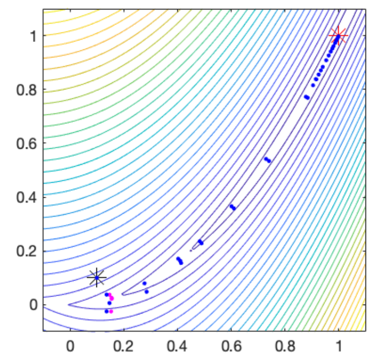



Figure 1. Contours of the nonsmooth Rosenbrock function (3) showing iterates generated by the gradient sampling method (blue dots) and the ordinary gradient method with the same line search (magenta dots). The black asterisk is the initial point and the red asterisk shows the unique minimizer.

The poor behavior of the gradient method in this context is well known, even in the convex case [HUL93]; see [AO18] for a discussion of the behavior of the gradient method on a simple nonsmooth convex function. In these experiments, for both algorithms, $x_1^2 - x_2$ was nonzero at all iterates, even in finite precision, so gradients were always defined. Figure 2 shows the function values $\{f(x^k)\}$ generated by the two methods. Both algorithms were terminated as soon as the objective and/or gradient was evaluated at 2000 points—including iterates, trial points in the line searches, and randomly generated points at which the gradient is evaluated for Algorithm 1. Algorithm 1 is able to reach iterates with much better objective values within the same budget.[6]

It is also instructive to consider a subgradient method [Sho85, Rus06] that sets iterates by

$$x^{k+1} \leftarrow x^k - t_k d^k,$$

[6]We used the following parameters for Algorithm 1: $\epsilon_0 = \nu_0 = 0.1$, $m = 3$, $\beta = 10^{-8}$, $\gamma = 0.5$, $\epsilon_{\text{opt}} = \nu_{\text{opt}} = 0$, and $\theta_\epsilon = \theta_\nu = 0.1$. The gradient method used the same line search with $\beta = 10^{-8}$ and $\gamma = 0.5$. Both algorithms used most of their function and gradient evaluations in later iterations. The final sampling radius for Algorithm 1 was $10^{-5}$.

where $d^k$ is any subgradient of $f$ at $x^k$ (i.e., any element of $\bar{\partial} f(x^k)$) and $\{t_k\}$ is set as a fixed stepsize or according to a diminishing stepsize schedule. This is a popular approach in the optimization literature, which has convergence guarantees in various contexts without requiring that the value of $f$ decreases at each iteration. By not requiring monotonic decrease, the method does not get stuck near the parabola on which $f$ is nonsmooth. However, progress is slow since the method has no mechanism for identifying the tangential direction of descent along the parabola. Instead, it is destined to oscillate back-and-forth across the parabola as it creeps tangentially toward the minimizer $x^*$. In this experiment, $x_1^2 - x_2$ was nonzero (even in finite precision) at all but a handful of the iterates, and since the only subgradient of $f$ at such a point is the gradient, the method is, for all practical purposes, identical to a gradient method with the same stepsizes. The iterates of this method with $\{t_k\} = \{0.1/k\}$ are shown in Figure 3, and the performance with different choices for $\{t_k\}$ is shown in Figure 4. With the same function and gradient evaluation budget as the methods above, this approach—for all stepsize choices—is slow. One might be able to obtain better results by tuning the stepsize choice further. Note, however, that Algorithm 1 does not require such parameter tuning.

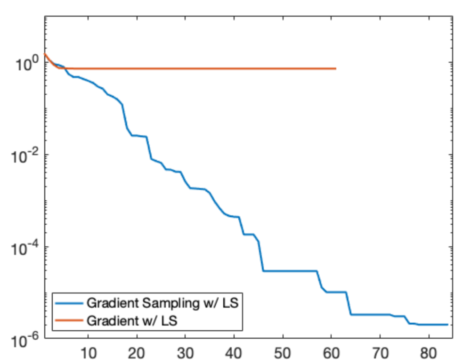



Figure 2. Function values by iteration number for the gradient sampling method and the gradient method, both run with a backtracking line search (LS).

Of course, there are other effective algorithms for nonsmooth optimization that we do not consider here, in part because they are significantly more complicated to describe; these include bundle methods [Kiw85, SZ92], which have been used extensively for decades, and quasi-Newton methods [LO13]. For a collection of surveys of recent developments in nonsmooth optimization methods, see [BGKM19].

The function $f$ in (3) is an example of an important class of functions, namely those that are *partly smooth* with respect to a manifold in the sense defined in [Lew02]. In the convex case, this concept is related to that of the $\mathcal{U}$-Lagrangian [LOS99].

# 3 Convergence Theory

Algorithm 1 is conceptually straightforward. At each iterate, one need only compute gradients at randomly sampled points, project the origin onto the convex hull of these gradients (by solving a strongly convex quadratic program (QP) for which specialized algorithms have been designed [Kiw86]), and perform a line search. The other details relate to dynamically setting the sampling radii $\{\epsilon_k\}$ and ensuring that the objective $f$ is differentiable at each iterate.

On the other hand, the convergence theory for the algorithm when minimizing a locally Lipschitz function involves important, subtle details. Rademacher's theorem states that locally Lipschitz functions are differentiable almost everywhere [Cla83], ensuring that the gradients sampled at the randomly generated points are well defined with probability one. However, this is not sufficient to ensure convergence. To obtain a satisfactory convergence result it is required that the set of points at which $f$ is *continuously differentiable* has full measure in $\mathbb{R}^n$. For further discussion of this issue, see [BCL$^+$19].

The following theorem, whose precise statement is taken from [BCL$^+$19], but whose proof depends on the convergence theorems in [BLO05, Kiw07], is the main convergence result for Algorithm 1. Other results of interest that can be proved relate to the behavior of the algorithm when the tolerances $\nu_{\text{opt}}$ and $\epsilon_{\text{opt}}$ are positive, so the algorithm terminates finitely, or when one sets $\theta_\epsilon = 1$, so that the sampling radius is fixed, in which case one can prove convergence to $\epsilon$-stationarity; see [BLO05, Kiw07, BCL$^+$19].

**Theorem 1.** *Suppose that $f$ is locally Lipschitz on $\mathbb{R}^n$ and continuously differentiable on an open set with full measure in $\mathbb{R}^n$. Suppose also that Algorithm 1 is run with $\nu_0 > 0$, $\nu_{\text{opt}} = \epsilon_{\text{opt}} = 0$, and strict reduction factors $\theta_\nu < 1$ and $\theta_\epsilon < 1$. Then, with probability one, Algorithm 1 is well defined in the sense that the sampled gradients exist in every iteration, the algorithm does not terminate, and either*

- $\{f(x^k)\} \searrow -\infty$ *or*
- $\{\nu_k\} \searrow 0$, $\{\epsilon_k\} \searrow 0$, *and each limit point $\bar{x}$ of the sequence $\{x^k\}$ is Clarke stationary for $f$, that is, $0 \in \bar{\partial} f(\bar{x})$.*

It has been shown that the result of Theorem 1 can be extended for some cases of non-locally Lipschitz $f$, in particular, when it is *directionally Lipschitz* [Lin09]. Extending it to the general non-locally Lipschitz setting, on the other hand, seems quite difficult. One can also prove that, in the case of minimizing finite-max functions, Algorithm 1 can achieve a linear rate of local convergence, at least in a certain probabilistic sense [HSS17]. This should not be too surprising given the connection between the GS methodology and standard steepest descent.

# 4 Enhancements

Since the inception and analysis of the initial GS algorithm in [BLO05], various enhancements and extensions

**Algorithm 1** : Gradient Sampling with a Line Search

**Require:** initial point $x^0$ at which $f$ is differentiable, initial sampling radius $\epsilon_0 \in (0, \infty)$, initial stationarity target $\nu_0 \in [0, \infty)$, sample size $m \geq n + 1$, line search parameters $(\beta, \gamma) \in (0, 1) \times (0, 1)$, termination tolerances $(\epsilon_{\text{opt}}, \nu_{\text{opt}}) \in [0, \infty) \times [0, \infty)$, and reduction factors $(\theta_\epsilon, \theta_\nu) \in (0, 1] \times (0, 1]$

1: **for** $k \in \mathbb{N}$ **do**

2: independently sample $\{x^{k,1}, \ldots, x^{k,m}\}$ uniformly from $\mathbb{B}(x^k, \epsilon_k) := \{x \in \mathbb{R}^n : \|x - x^k\|_2 \leq \epsilon_k\}$

3: compute $g^k$ as the solution of $\min_{g \in \mathcal{G}^k} \frac{1}{2}\|g\|_2^2$, where $\mathcal{G}^k := \text{conv}\{\nabla f(x^k), \nabla f(x^{k,1}), \ldots, \nabla f(x^{k,m})\}$

4: **if** $\|g^k\|_2 \leq \nu_{\text{opt}}$ and $\epsilon_k \leq \epsilon_{\text{opt}}$ **then** terminate

5: **if** $\|g^k\|_2 \leq \nu_k$

6: **then** set $\nu_{k+1} \leftarrow \theta_\nu \nu_k$, $\epsilon_{k+1} \leftarrow \theta_\epsilon \epsilon_k$, and $t_k \leftarrow 0$

7: **else** set $\nu_{k+1} \leftarrow \nu_k$, $\epsilon_{k+1} \leftarrow \epsilon_k$, and

$$t_k \leftarrow \max\left\{t \in \{1, \gamma, \gamma^2, \ldots\} : f(x^k - tg^k) < f(x^k) - \beta t \|g^k\|_2^2\right\} \tag{4}$$

8: **if** $f$ is differentiable at $x^k - t_k g^k$

9: **then** set $x^{k+1} \leftarrow x^k - t_k g^k$

10: **else** set $x^{k+1}$ randomly as any point where $f$ is differentiable such that

$$f(x^{k+1}) < f(x^k) - \beta t_k \|g^k\|_2^2 \ \text{ and } \ \|x^k - t_k g^k - x^{k+1}\|_2 \leq \min\{t_k, \epsilon_k\}\|g^k\|_2 \tag{5}$$

11: **end for**

have appeared in the literature. First, a few fundamental advances were published in [Kiw07]; in particular, in this work, Kiwiel showed how to simplify the analysis of a basic GS algorithm and extend it for some interesting algorithm variants, such as when invoking a trust region methodology. Other proposed enhancements include techniques for avoiding the differentiability check in Steps 8–10 of Algorithm 1) [Kiw07, HSS16], performing the gradient sampling adaptively so that only $\mathcal{O}(1)$ gradients need to be sampled in each iteration [CQ13, CQ15], and for incorporating second-order derivatives or approximations, say by borrowing quasi-Newton ideas from the smooth optimization literature [CQ13, CQ15]. Added benefits of adaptive sampling are that one can re-use gradients computed in previous iterations and *warm-start* the solve of each QP so that the computation of each search direction becomes relatively inexpensive.

The GS methodology has also been extended for solving constrained optimization problems. Specifically, a Riemannian GS method has been proposed for optimization on manifolds [HU17], and a so-called SQP-GS method, which merges the GS methodology with that of a penalty sequential quadratic programming (SQP) technique from the smooth optimization literature, has been proposed for solving inequality constrained optimization problems in which the objective and constraint functions may be nonsmooth and/or nonconvex [CO12]. A feasible variant of the SQP-GS method has also been proposed, which establishes a path for the design of two-phase approaches: a first phase seeking feasibility and a second phase seeking optimality [TLJL14].

Another interesting line of work has been on adaptions of the GS methodology for designing derivative-free algorithms for minimizing nonsmooth functions. In a couple of these cases, authors have proposed to use the GS methodology in a relatively straightforward manner with gradients replaced by gradient approximations constructed using function evaluations [Kiw10, HN13]. There has also been work on methods that do not borrow the gradient sampling strategy *per se*, but are still motivated by the GS methodology in terms of the types of subproblems that are employed to compute search directions [LMW16].

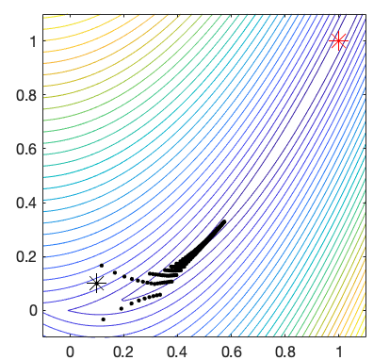



Figure 3. Contours of the nonsmooth Rosenbrock function (3) showing iterates generated by the subgradient method with $\{t_k\} = \{0.1/k\}$. The black asterisk is the initial point and the red asterisk shows the unique minimizer.

For more information on the enhancements and extensions that have been made to the GS methodology over the past decade, as well as information about available software and success stories in practice, see [BCL+19].

# 5 Closing Remarks

Gradient sampling is a conceptually straightforward, yet powerful approach for extending the steepest descent methodology to the minimization of nonsmooth, nonconvex functions. The fundamental idea of GS is to stabilize the notion of a steepest descent direction by finding the minimum norm element of the convex hull of gradients evaluated at points randomly sampled near the current iterate. The methodology enjoys a solid theoretical foundation and has been enhanced and extended in various ways, such as for solving constrained optimization problems.

There remain various interesting avenues of research related to the GS methodology. For example, it remains an open question how far one may be able to extend the convergence theory for a GS method in terms of minimizing non-locally Lipschitz functions; e.g., can one extend the GS theory for the class of semi-algebraic, but not locally Lipschitz or directionally Lipschitz functions? On the other hand, one can imagine various opportunities for exploring tailored GS approaches when one aims to minimize a function for which one has knowledge about the structure of the nonsmoothness of a function $f$. How should sampling be performed when, at any given iterate, one has knowledge about the directions in which $f$ is smooth and directions in which it is nonsmooth (at least in a neighborhood of the current iterate)?

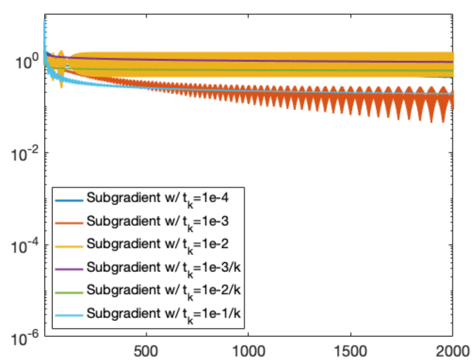



Figure 4. Function values by iteration number for the subgradient method with different stepsize sequences, indicated by the formula for $t_k$ in the legend.

One also has the sense that there remain numerous avenues to pursue in the context of constrained optimization. Given the range of methodologies for solving smooth constrained optimization problems, one could explore techniques that combine these approaches with gradient sampling so that convergence guarantees could potentially be obtained when handling nonsmooth functions as well. One might also re-evaluate the use of certain methods, such as some exact penalty methods, which have previously fallen out of favor due to the presence of nonsmoothness. After all, the issues that inhibited the effectiveness of such approaches might no longer be of concern since GS might naturally overcome them.

Finally, there remain various open questions about the possible connections between the GS methodology and other randomized and/or stochastic optimization methods. The basic GS method involves computing the minimum norm element in the convex hull of gradients evaluated at randomly generated points. Can the GS theory be extended when the subproblems for computing the search directions are only solved approximately? If so, this might represent a step toward tying the convergence theory of GS with those of other randomized/stochastic gradient/subgradient approaches, which have attracted a lot of recent attention; see, e.g., [DD18, DDKL19].

## Acknowledgment


The authors would like to sincerely thank the ICS Prize Committee, which was chaired by Andreas Wächter and included Fatma Kilinc-Karzan, Douglas Shier, and Cole Smith. The prize was awarded for the articles [BLO05, CMO17, CO12, CQ13, CQ15, BCL+19], which involved co-authorship by Tim Mitchell, Xiaocun Que and Lucas E. A. Simões. We are honored to receive this prize and are grateful for this opportunity to promote work related to the GS methodology. James V. Burke is supported in part by the U.S. National Science Foundation grant DMS-1514559, Frank E. Curtis is supported in part by the U.S. Department of Energy grant DE-SC0010615 and National Science Foundation grant CCF-1740796, Adrian S. Lewis is supported in part by the U.S. National Science Foundation grant DMS-1613996, and Michael L. Overton is supported in part by the U.S. National Science Foundation grant DMS-1620083.


## References


[AO18] A. Asl and M. L. Overton. Analysis of the Gradient Method with an Armijo-Wolfe Line Search on a Class of Nonsmooth Convex Functions, 2018. arXiv:1711.08517v2.

[BCL+19] J. V. Burke, F. E. Curtis, A. S. Lewis, M. L. Overton, and L. E. A. Simões. Gradient Sampling Methods for Nonsmooth Optimization. In A. Bagirov, M. Gaudioso, N. Karmitsa, and M. Mäkelä, editors, *Numerical Nonsmooth Optimization*. Springer, 2019. arXiv:1804.11003.

[BGKM19] A. Bagirov, M. Gaudioso, N. Karmitsa, and M.M. Mäkelä, editors. *Numerical Nonsmooth Optimization*. Springer, 2019.

[BLO02] J. V. Burke, A. S. Lewis, and M. L. Overton. Approximating Subdifferentials by Random Sampling of Gradients. *Mathematics of Operations Research*, 27(3):567–584, 2002.

[BLO05] J. V. Burke, A. S. Lewis, and M. L. Overton. A Robust Gradient Sampling Algorithm for Nonsmooth, Nonconvex Optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[Cla75] F. H. Clarke. Generalized Gradients and Applications. *Transactions of the American Mathematical Society*, 205:247–262, 1975.

[Cla83] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley, New York, 1983. Reprinted by SIAM, Philadelphia, 1990.

[CMO17] F. E. Curtis, T. Mitchell, and M. L. Overton. A BFGS-SQP method for Nonsmooth, Nonconvex, Constrained Optimization and its Evaluation using Relative Minimization Profiles. *Optimization Methods and Software*, 32(1):148–181, 2017.

[CO12] F. E. Curtis and M. L. Overton. A Sequential Quadratic Programming Algorithm for Nonconvex, Nonsmooth Constrained Optimization. *SIAM Journal on Optimization*, 22(2):474–500, 2012.

[CQ13] F. E. Curtis and X. Que. An Adaptive Gradient Sampling Algorithm for Nonsmooth Optimization. *Optimization Methods and Software*, 28(6):1302–1324, 2013.

[CQ15] F. E. Curtis and X. Que. A Quasi-Newton Algorithm for Nonconvex, Nonsmooth Optimization with Global Convergence Guarantees. *Mathematical Programming Computation*, 7(4):399–428, 2015.

[DD18] D. Davis and D. Drusvyatskiy. Stochastic Model-Based Minimization of Weakly Convex Functions, 2018. arXiv:1803.06523, to appear in *SIAM Journal on Optimization*.

[DDKL19] D. Davis, D. Drusvyatskiy, S. Kakade, and J. D. Lee. Stochastic Subgradient Method Converges on Tame Functions, 2019. arXiv:1804.07795, to appear in *Foundations of Computational Mathematics*.

[Gol77] A. A. Goldstein. Optimization of Lipschitz Continuous Functions. *Mathematical Programming*, 13(1):14–22, 1977.

[HN13] W. Hare and J. Nutini. A Derivative-Free Approximate Gradient Sampling Algorithm for Finite Minimax Problems. *Computational Optimization and Applications*, 56(1):1–38, 2013.

[HSS16] E. S. Helou, S. A. Santos, and L. E. A. Simões. On the Differentiability Check in Gradient Sampling Methods. *Optimization Methods and Software*, 31(5):983–1007, 2016.

[HSS17] E. S. Helou, S. A. Santos, and L. E. A. Simões. On the Local Convergence Analysis of the Gradient Sampling Method for Finite Max-Functions. *Journal of Optimization Theory and Applications*, 175(1):137–157, 2017.

[HU17] S. Hosseini and A. Uschmajew. A Riemannian Gradient Sampling Algorithm for Nonsmooth Optimization on Manifolds. *SIAM Journal on Optimization*, 27(1):173–189, 2017.

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*, volume 305 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1993.

[Kiw85] K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics. Springer-Verlag, New York, NY, USA, 1985.

[Kiw86] K. C. Kiwiel. A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization. *IMA Journal of Numerical Analysis*, 6(2):137–152, 1986.

[Kiw07] K. C. Kiwiel. Convergence of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[Kiw10] K. C. Kiwiel. A Nonderivative Version of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 20(4):1983–1994, 2010.

[Lew02] A. S. Lewis. Active Sets, Nonsmoothness, and Sensitivity. *SIAM Journal on Optimization*, 13(3):702–725, 2002.

[Lin09] Q. Lin. *Sparsity and Nonconvex Nonsmooth Optimization*. PhD thesis, Department of Mathematics, University of Washington, 2009.

[LMW16] J. Larson, M. Menickelly, and S. M. Wild. Manifold Sampling for $\ell_1$ Nonconvex Optimization. *SIAM Journal on Optimization*, 26(4):2540–2563, 2016.

[LO13] A. S. Lewis and M. L. Overton. Nonsmooth Optimization via Quasi-Newton Methods. *Mathematical Programming*, 141(1-2):135–163, 2013.

[LOS99] C. Lemaréchal, F. Oustry, and C. Sagastizábal. The $\mathcal{U}$-lagrangian of a convex function. *Transactions of the American Mathematical Society*, 352(2):711–729, 1999.

[Ros60] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, 1960.

[Rus06] A. Ruszczyński. *Nonlinear Optimization*. Princeton University Press, Princeton, NJ, 2006.

[Sho85] N.Z. Shor. *Minimization Methods for Non-Differentiable Functions*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin, Heidelberg, Germany, 1985.

[SZ92] H. Schramm and J. Zowe. A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.

[TLJL14] C.-M. Tang, S. Liu, J.-B. Jian, and J.-L. Li. A Feasible SQP-GS Algorithm for Nonconvex, Nonsmooth Constrained Optimization. *Numerical Algorithms*, 65(1):1–22, 2014.

# Research Highlight: Asymptotically Optimal Exact Solution of Sparse Linear Systems via Left-Looking Roundoff-Error-Free LU Factorization

by Christopher Lourenco[7]

We would like to thank the Informs Computing Society student paper award committee members, Sergiy Butenko and Frank E. Curtis, and chair, Anna Nagurney, for the honor of being awarded honorable mention in the ICS Student Paper Award competition.

This research focuses on exact linear programming (LP). LP is widely considered a solved problem; however, there exists a class of problems for which commercial LP solvers are inadequate. Indeed, commercial optimization solvers come with limited guarantees: they may classify suboptimal solutions as optimal or even feasible bases as infeasible and vice versa [3, 11]. While it is true that commercial solvers produce excellent solutions for a majority of instances, a number of applications—including feasibility problems, compiler optimization, computational based mathematical proofs, and numerically unstable instances [9, 16, 10]—generate underlying LPs which require exact solutions. Ill-conditioned instances are of particular interest, as 72% of the real world LPs within the NETLIB LP repository are ill-conditioned [14]; which leads to the conjecture that ill-conditioned LPs are frequent in practice [13]. However, solving LPs exactly is challenging due to the roundoff errors intrinsic to floating point arithmetic. While they are individually harmless, roundoff errors may propagate, accumulate, and magnify, ultimately rendering the output of linear solvers egregiously incorrect. It is imperative, therefore, that efficient algorithms and robust software exist which account for, and ultimately eliminate, roundoff errors when solving LPs.

Currently, there are two approaches for solving LPs exactly: certify and repair [3, 1] and LP iterative refinement [8, 7]. Each method employs a mixed precision approach in which the majority of operations is performed in floating point precision. Both guarantee exactness by solving—via full precision rational arithmetic LU factorizations[8]—the *sparse* linear systems associated with promising basis matrices. However, these full precision rational-arithmetic LU factorizations often serve as the bottleneck of exact LP solvers, as they may occupy up to 90% of the run time of exact LP algorithms [7]. Despite their prevalent use, very little research has been devoted to improving exact LU factorizations.

To address the drawbacks of rational LU factorization, the (dense) roundoff-error-free (REF) LU factorization was developed. The REF LU factorization is based on integer-preserving Gaussian elimination (IPGE) and exactly solves a linear system using exclusively integer arithmetic with the added property that the bit-length of each entry is bounded polynomially [4]. This polynomial bound is a key property of REF LU, as rational LU factorization approaches achieve these bounds only via computationally costly greatest common divisor operations [15]. Associated computational tests showed that REF LU outperformed dense rational LU factorization by one order of magnitude in run time while requiring half the memory [5].

This research expands REF LU to the sparse setting by developing the theoretical and computational framework for the Sparse Left-looking Integer-Preserving (SLIP LU) factorization. SLIP LU, the first fully sparse integer-preserving LU factorization, computes the sparse factorization, $A = LDU$, one column at a time. At iteration $k$, SLIP LU computes the $k$th column of $L$ and $U$ via a two step process. First, symbolically, SLIP LU determines the nonzero pattern of the $k$th column of $L$ and $U$ via a graph traversal algorithm, where a sequence of depth first searches is performed on the graph of the matrix $L$. Second, numerically, we derive a new algorithm which combines left-looking LU factorization [6], sparse extensions to IPGE [12], and REF forward substitution [4] so that the $L$ and $U$ factors are computed using only the necessary IPGE operations. Note that, though the factorization is $A = LDU$, only the integral $L$ and $U$ matrices are required to solve the linear system $A\mathbf{x} = \mathbf{b}$. In addition, due to the special structure of the derived algorithm, we prove that the computational complexity of SLIP LU is proportional to the cost of the factorization's arithmetic work, meaning that SLIP LU is asymptotically optimal with respect to solving sparse linear systems. In practice, this property is difficult to achieve, as in floating point arithmetic, left-looking LU factorization is the only algorithm to solve sparse linear systems in time proportional to arithmetic work [6]. Likewise, to our knowledge, SLIP LU is the only exact method to solve sparse linear systems with this property.

Computationally, we implemented the SLIP LU factorization in C++ and compared it to a modern left-looking rational-arithmetic LU factorization (denoted QLU). We performed this computational comparison on the `BasisLIB` repository, a test set of 276 real-world LP basis matrices received as output from the `QSopt` LP solver [2]. This repository serves as a good test case for exact LP, as 51% of the matrices within it have an estimated condition number exceeding $10^8$, meaning that the underlying linear systems are prone to numerical errors. In terms of factorization construction time, SLIP LU is faster than QLU for 59% of the instances while having an average and geometric mean 6.1 and 1.5 times smaller than those of QLU, respectively. More strikingly, in terms of forward and backward substitution, SLIP LU is faster for 75% of the instances while having an average and geometric

[7] Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX. `clouren@tamu.edu`. This paper was submitted to the SIAM Journal on Matrix Analysis and Applications (SIMAX) and is currently under revision.

[8] an LU factorization factors the matrix $A$ into the product of a lower triangular, $L$, and upper triangular, $U$, matrix, $A = LU$

mean 3.4 and 4.0 times smaller than those of QLU. Graphically, these results are illustrated via the performance profiles in Figures 5 and 6. Altogether, these results offer compelling evidence that SLIP LU dramatically outperforms rational left-looking LU factorization for solving real world, unsymmetric sparse linear systems. Specifically, SLIP LU is shown to perform well on those real-world linear systems encountered in exact LP. The code associated with the SLIP LU factorization is publicly available at https://github.com/clouren/SLIP_LU.

Figure 5. SLIP LU Dominates QLU in Factorization Time

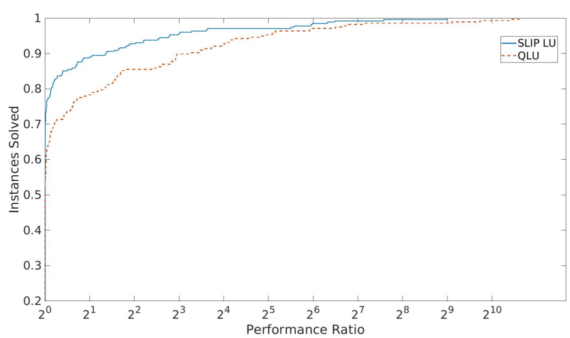
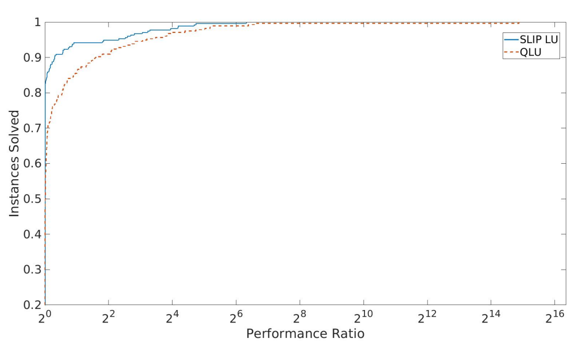



Figure 6. SLIP LU Dominates QLU in Substitution Times




## References


[1] D. L. Applegate, W. Cook, S. Dash, and D. G. Espinoza, *Exact solutions to linear programming problems*, Operations Research Letters, 35 (2007), pp. 693–699.

[2] W. Cook and D. E. Steffy, *Solving very sparse rational systems of equations*, ACM Transactions on Mathematical Software (TOMS), 37 (2011), p. 39.

[3] M. Dhiflaoui, S. Funke, C. Kwappik, K. Mehlhorn, M. Seel, E. Schömer, R. Schulte, and D. Weber, *Certifying and repairing solutions to large lps how good are lp-solvers?*, in Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2003, pp. 255–256.

[4] A. R. Escobedo and E. Moreno-Centeno, *Roundoff-error-free algorithms for solving linear systems via cholesky and lu factorizations*, INFORMS Journal on Computing, 27 (2015), pp. 677–689.

[5] A. R. Escobedo, E. Moreno-Centeno, and C. Lourenco, *Solution of dense linear systems via roundoff-error-free factorization algorithms: Theoretical connections and computational comparisons*, ACM Transactions on Mathematical Software (TOMS), 44 (2018), p. 40.

[6] J. R. Gilbert and T. Peierls, *Sparse partial pivoting in time proportional to arithmetic operations*, SIAM Journal on Scientific and Statistical Computing, 9 (1988), pp. 862–874.

[7] A. M. Gleixner, *Exact and fast algorithms for mixed-integer nonlinear programming*, (2015).

[8] A. M. Gleixner, D. E. Steffy, and K. Wolter, *Iterative refinement for linear programming*, INFORMS Journal on Computing, 28 (2016), pp. 449–464.

[9] T. C. Hales, *A proof of the kepler conjecture*, Annals of mathematics, (2005), pp. 1065–1185.

[10] E. Klotz, *Identification, assessment, and correction of ill-conditioning and numerical instability in linear and integer programs*, in Bridging Data and Decisions, INFORMS, 2014, pp. 54–108.

[11] T. Koch, *The final netlib-lp results*, Operations Research Letters, 32 (2004), pp. 138–142.

[12] H. R. Lee and B. D. Saunders, *Fraction free gaussian elimination for sparse matrices*, Journal of symbolic computation, 19 (1995), pp. 393–402.

[13] A. Neumaier and O. Shcherbina, *Safe bounds in linear and mixed-integer linear programming*, Mathematical Programming, 99 (2004), pp. 283–296.

[14] F. Ordóñez and R. M. Freund, *Computational experience and the explanatory value of condition measures for linear optimization*, SIAM Journal on Optimization, 14 (2003), pp. 307–333.

[15] A. Schrijver, *Theory of linear and integer programming*, John Wiley & Sons, 1998.

[16] D. E. Steffy, *Topics in exact precision mathematical programming*, PhD thesis, Georgia Institute of Technology, 2011.

# $\mathcal{V}$-Polyhedral Disjunctive Cuts

Aleksandr M. Kazachkov*

*It is an honor to receive the ICS Student Paper Award. Thank you to the committee, chaired by Anna Nagurney and including Sergiy Butenko and Frank Curtis, for selecting our paper.*

## 1 Introduction

Our paper presents a new framework for efficiently and *non-recursively* generating a large number of strong disjunctive *cutting planes*, or *cuts*. We are motivated by a drawback of many existing techniques, in their reliance on recursion to reach strong cuts, i.e., on computing cuts from previously-derived ones. This can result in numerical issues (e.g., due to compounding inaccuracies) and a "tailing off" of the strength of the cuts in later rounds [5, 17]. Our goal is to circumvent recursion without sacrificing strength.

One way to accomplish this is by generating cuts through stronger *disjunctions* (compared to those that are commonly used), which partition $\mathbb{R}^n$ such that the integer-feasible region, denoted $P_I$, is contained in the union of the disjunctive terms. Concretely, a disjunction takes the form $\bigvee_{t\in\mathcal{T}}\{x \in \mathbb{R}^n : D^t x \geq D^t_0\}$. where $\mathcal{T}$ is a finite index set. With $P := \{x \in \mathbb{R}^n : Ax \geq b\}$ and $P_I := \{x \in P : x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}$, $\mathcal{I} \subseteq \{1,\dots,n\}$, we denote *disjunctive term* $t \in \mathcal{T}$ by $P^t := \{x \in P : D^t x \geq D^t_0\}$. Let $P_D := \operatorname{cl}\operatorname{conv}(\cup_{t\in\mathcal{T}} P^t)$ be the *disjunctive hull*, the *closed convex hull* of the points of $P$ satisfying the disjunction. For validity, we assume the disjunction satisfies $P_I \subseteq P_D$ and $\bar{x} \notin P_D$, where $\bar{x}$ is an optimal solution to the linear programming relaxation $\min_x\{c^\mathsf{T} x : x \in P\}$.

Although producing cuts from stronger disjunctions avoids recursion, the challenge becomes how to do so efficiently as $|\mathcal{T}|$ grows. The prevailing method for generating disjunctive cuts, which was introduced by Balas [2] and has come to be known as *lift-and-project* [3], quickly becomes too costly to use in practice due to the higher-dimensional representation it employs. The innovation of our framework is a cut generation scheme formulated in the original dimension ($n$) of the problem via an efficient use of the $\mathcal{V}$*-polyhedral* perspective, representing a polyhedron through its extreme points and rays, in contrast to the inequality description underlying lift-and-project.

*Based on joint work with Egon Balas [4].

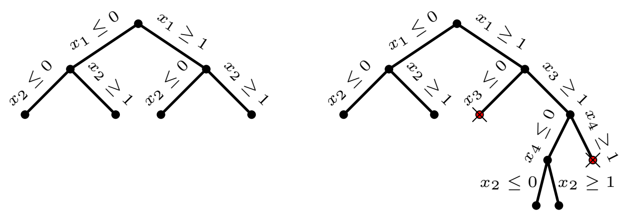
## 2 Point-Ray Linear Program

Let $\mathcal{P}$ and $\mathcal{R}$ denote sets of points and rays in $\mathbb{R}^n$. Define the *point-ray linear program* (PRLP), taking the point-ray collection $(\mathcal{P},\mathcal{R})$ and an objective direction $w \in \mathbb{R}^n$ as an input, as follows:

$$\begin{aligned} \min_{\alpha,\beta} \quad & \alpha^\mathsf{T} w \\ & \alpha^\mathsf{T} p \geq \beta && \text{for all } p \in \mathcal{P} \\ & \alpha^\mathsf{T} r \geq 0 && \text{for all } r \in \mathcal{R}. \end{aligned} \tag{PRLP}$$

The feasible solutions $(\alpha,\beta)$ to (PRLP) correspond to inequalities $\alpha^\mathsf{T} x \geq \beta$ that we call $\mathcal{V}$*-polyhedral cuts* (VPCs). Define the *point-ray hull* as $\operatorname{conv}(\mathcal{P})+\operatorname{cone}(\mathcal{R})$. Theorem 1 shows that the extreme ray solutions to (PRLP) correspond to facet-defining inequalities for the point-ray hull.

**Theorem 1.** *The inequality $\alpha^\mathsf{T} x \geq \beta$ is valid for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ if and only if $(\alpha,\beta)$ is a feasible solution to (PRLP). The inequality defines a facet of the point-ray hull if and only if the solution $(\alpha,\beta)$ is an extreme ray of (PRLP).*




(a) The disjunction is all possible assignments of $x_1$ and $x_2$.

(b) Branching on different variables results in two stronger disjunctive terms.

Figure 1: Two four-term disjunctions (the leaf nodes of the trees).

We now describe the steps that are needed to generate cuts from (PRLP) in practice. This involves (1) choosing a disjunction, (2) selecting the point-ray collection, and (3) deciding the objective directions. There are many nontrivial options for each of these parts, so we support our choices with theoretical results.

## 2.1 Choosing strong disjunctions

Most cutting plane research focuses on generating cuts from "shallow" disjunctions: those that utilize relatively few (one or two) integer variables at a time, such as split or cross disjunctions. Strong cuts are obtained by a combination of recursion and considering several shallow disjunctions simultaneously in each round. We follow a different paradigm, in which we expend extra effort to find one strong "deep" disjunction, with the hope that this effort leads to better cuts than those from multiple shallow disjunctions and many rounds.

Specifically, our disjunctions come from the leaf nodes of a partial branch-and-bound tree. The partial tree may be asymmetric and include pruning by infeasibility, integrality, and bound. We demonstrate this in Figure 1, contrasting a cross disjunction generated from two integer variables $x_1$ and $x_2$ to a four-term disjunction that might be obtained using the branch-and-bound process. Cuts from partial branch-and-bound trees have been used in several contexts in the past; we refer the interested reader to [12] for coverage of related literature.

## 2.2 Normalization of the PRLP

The PRLP we use in our experiments differs from (PRLP) in two ways. First, to get extreme point solutions, we truncate the cone defining the feasible region of (PRLP). We choose to normalize (PRLP) by fixing $\beta = 1$. Second, we formulate (PRLP) in the *nonbasic space* defined by the cobasis at $\bar{x}$. In this space and with $\beta = 1$, every basic feasible solution $\alpha$ to (PRLP) corresponds to an inequality $\alpha^\mathsf{T} x \geq 1$ violated by $\bar{x}$.

## 2.3 Proper point-ray collections

One challenge with a $\mathcal{V}$-polyhedral representation is that the number of constraints of (PRLP) can grow exponentially large with respect to the original formulation size of $P$, and these rows are necessary in that dropping them may yield invalid cuts. For this reason, prior related work resorts to row generation to guarantee validity [16, 18]. We instead show that the expensive row generation can be avoided via a properly chosen compact collection of points and rays, which will suffice to produce valid cuts, albeit a subset of the entire pool of possible disjunctive cuts.

To guarantee validity of the cuts, we adapt the definition of a *proper* point-ray collection from [13].

**Definition 2.** *The point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is called* proper *if* $\alpha^\mathsf{T} x \geq \beta$ *is valid for* $P_I$ *whenever* $(\alpha, \beta)$ *is feasible to* (PRLP).

As a direct corollary to Theorem 1, we obtain a necessary and sufficient condition for a point-ray collection to be proper.

**Corollary 3.** *A point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is proper if and only if* $P_I \subseteq \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$.

Of course, we do not work with the integer hull directly. The intermediary is the disjunctive hull. The next corollary is a key result for the development of a practical procedure working with points and rays. It states that as long as the point-ray hull forms a $\mathcal{V}$*-polyhedral relaxation* of $P_D$, then the point-ray collection is proper.

**Corollary 4.** *A point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is proper if* $P^t \subseteq \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ *for all* $t \in \mathcal{T}$, *or, equivalently, if* $P_D \subseteq \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$.

## 2.4 Simple cone relaxations

Instead of pursuing all facet-defining inequalities for the disjunctive hull, we use a relaxation of $P_D$ with a compact $\mathcal{V}$-polyhedral description. One convenient relaxation for each disjunctive term is the *basis cone* $C^t$ at an optimal solution $p^t$ to $\min_x\{c^\mathsf{T} x : x \in P^t\}$, formed from $p^t$ and a *cobasis* $\mathcal{N}(p^t)$ associated with $p^t$. This cone is defined as the intersection of the $n$ inequalities corresponding to the nonbasic variables indexed by $\mathcal{N}(p^t)$, and it has a compact $\mathcal{V}$-polyhedral description, with only one extreme point ($p^t$) and $n$ extreme rays. We refer to the union of these points and rays across all terms as the *simple point-ray collection* $(\mathcal{P}^0, \mathcal{R}^0)$, and we will use the shorthand $P_D^0 := \text{conv}(\mathcal{P}^0) + \text{cone}(\mathcal{R}^0)$ to denote the corresponding point-ray hull. Define *PRLP$^0$* as (PRLP) with $(\mathcal{P}, \mathcal{R}) = (\mathcal{P}^0, \mathcal{R}^0)$. The cuts from PRLP$^0$ will be called *simple* VPCs. We state their validity as Theorem 5.

**Theorem 5.** *The simple point-ray collection* $(\mathcal{P}^0, \mathcal{R}^0)$ *is proper.*

The feasible region of PRLP$^0$ has the same number of constraints as the linear program for generating lift-and-project cuts, but it only has $n$ variables (whereas for lift-and-project, the number of variables increases with $|T|$).

## 2.5 VPCs corresponding to facets of the disjunctive hull

Though $P_D^0$ is a drastic relaxation of $P_D$, in that it is defined by a small fraction of the inequalities defining $P_D$, we show that $P_D^0$ can be a very mild relaxation in the region of $P_D$ of interest to us. It is clear that not all facets of $P_D$ are captured by $P_D^0$; see, for example, facet $F_1$ in Figure 2. The figure also illustrates another phenomenon, which is the existence of what we call *stray rays*. Although a ray is added to the point-ray collection from some particular point (when building the point-ray collection), it is ultimately added to *all* points in $\mathcal{P}^0$ to calculate $P_D^0$. This can create facets of $P_D^0$ that are significantly weaker than facets of $P_D$. A ray is stray when it is tight for a facet of $P_D^0$, but when the point that the ray originated from is not tight for the facet. Thus, ray $r$ in Figure 2 is stray.

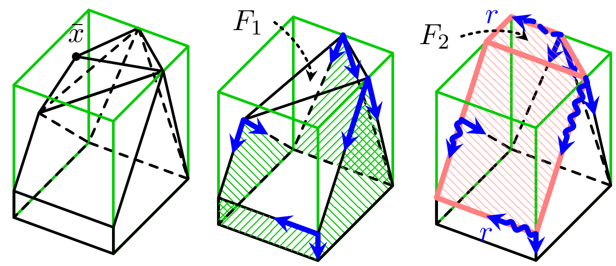



Figure 2: Stray rays can create unexpected facets of $P_D^0$.

From this, we can also define two types of facets of $P_D^0$. We say a facet of $P_D^0$ is *standard* if there is a corresponding basis of PRLP$^0$ with no stray rays. Otherwise, the facet is *nonstandard*. We apply this in Theorem 6 to state a sufficient condition for a facet of the point-ray hull to be a facet of the disjunctive hull.

**Theorem 6.** *Suppose the basis defining* $p^t$ *is unique for each* $t \in \mathcal{T}$*. If a facet of* $P_D^0$ *is standard, then it is a facet of* $P_D$ *that cuts* $\bar{x}$*.*

In the paper, we give special attention to the case of a split disjunction (given its importance in prior work), and show that all simple VPCs define facets of $P_D$ when there is no degeneracy.

## 2.6 Choosing appropriate objectives

Having set up the constraints of the PRLP, we need to decide which objective directions $w$ to pursue. Choosing these carefully is critical to the success of any VPC algorithm, as the objectives directly determine the nature of the VPCs obtained. Moreover, it is imperative to make the cut-generating process efficient, to prevent "failures" that could occur: PRLP$^0$ can be (1) infeasible, (2) feasible but unbounded, or (3) feasible, bounded, but the cut for a particular $w$ is a duplicate of a previously generated cut. Failure (1), for instance, occurs if $\bar{x}$ belongs to $P_D^0$.

Our paper contains the details of how we are able to avoid failures of types (1) and (2) (and drastically reduce the third type of failures). We merely mention here a key difference in our implementation from the typical viewpoint when generating cuts. Namely, the usual approach is to find cuts with high violation with respect to $\bar{x}$, or another point not in $P_D$. However, this classic paradigm may overemphasize cutting irrelevant parts of the relaxation (see, e.g., the discussion in [8]). The alternative we pursue is to find cuts that minimize the slack on points that do belong to $P_D$, through which we can also utilize more structural information about the disjunctive hull, readily available for us due to our use of a $\mathcal{V}$-polyhedral representation.

| Set | # inst | | G | V | V+G | GurF | V+GurF | GurL | V+GurL |
|---|---|---|---|---|---|---|---|---|---|
| All | 184 | Avg % | 17.3 | 15.6 | 27.0 | 26.0 | 33.0 | 46.5 | 52.1 |
| | | Wins | | 91 | 156 | | 143 | | 116 |
| ≥10% | 87 | Avg % | 14.4 | 29.6 | 33.5 | 20.0 | 32.6 | 38.8 | 50.0 |
| | | Wins | | 71 | 84 | | 73 | | 68 |

Table 1: Percent gap closed by VPCs. "All" is the set of instances with $c^\mathsf{T} p^* > c^\mathsf{T}\bar{x}$. The set "≥10%" contains all instances for which VPCs close at least 10% of the gap. "G", "V", "GurF", and "GurL" refer to the GMICs, VPCs, and the first and last round of cuts at `Gurobi`'s root node.

# 3 Computational Results

In this section, we briefly summarize our computational experience with VPCs. The goals of the experiments are to assess (1) the strength of VPCs by the percent root gap closed by one round of the cuts, and (2) the effectiveness of VPCs when added at the root and used as part of branch-and-bound.

We test 6 different disjunctions, defined by $\ell \in \{2, 4, 8, 16, 32, 64\}$ leaf nodes of a partial branch-and-bound tree generated by default `Cbc` settings. We use one round of rank one cuts. The cut limit is set to $|\{j \in \mathcal{I} : \bar{x}_j \notin \mathbb{Z}\}|$. All algorithms are implemented in C++ in the COIN-OR framework [15] using `Clp 1.16` and `Cbc 2.9`. We test the effectiveness of VPCs within branch-and-bound by adding them as user cuts within `Gurobi 7.5` [11]. The instances we select are among those with at most 5,000 rows and columns from the MIPLIB [1, 6, 7, 14], CORAL [9], and NEOS test sets. Every instance is preprocessed by `Gurobi`'s presolve.

**Percent integrality gap closed.** We measure percent integrality gap closed on the 184 instances for which the disjunctive lower bound is strictly greater than $c^\mathsf{T}\bar{x}$. The results (in Table 1) indicate the strength of VPCs, compared to the baseline of Gomory mixed-integer cuts (GMICs) [10] and the default cuts in `Gurobi`. Namely, using VPCs and GMICs together leads the average percent gap closed at the root to increase from 17% to 27%, with more gap closed on 156 of the 184 instances. Within `Gurobi`, the cuts continue to be very strong. For the first round of cuts at the root, the percent gap closed goes from 25% (without VPCs) to 33% (with them), with strictly better outcomes for 143 of the 184 instances. For the last round of cuts at the root, the percent gap closed increases from 46.5% to 52% by using VPCs.

An important conclusion we can draw is that our procedure can help avoid the "tailing-off" effect from recursive applications of cuts: without requiring recursion, by simply using a (sufficiently) stronger disjunction, we make steady progress (on average) toward the optimal value of (IP). However, this is only in terms of percent gap closed; as we discuss next, the story when using the cuts within branch-and-bound is completely different, in which seemingly weaker cuts may lead to better performance.

**Branch-and-bound effect.** We now turn to the second metric, of the effect of our cuts on branch-and-bound in terms of time and number of nodes when VPCs are added as user cuts to `Gurobi`, where (in the paper) we report the fastest solution time by `Gurobi` across the six different partial trees tested per instance, but also including the time from `Gurobi` run without VPCs as one of the possible minima, indicating the option of not using VPCs for an instance.

The results reported in the paper show unequivocally that using VPCs in this way vastly dominates `Gurobi` run with one random seed, yielding a drastic reduction in the average number of seconds and nodes to solve each instance. Of course, in this comparison, by definition, the time reported for VPCs weakly dominates that of `Gurobi`. The purpose of such a comparison is showing the theoretical benefit of the VPC approach: with an ideal way to select a partial tree per instance (including knowing when not to use VPCs at all), these are the results one would see. Unfortunately, in practice, we do not have access to such an oracle, but the results suggest the need for crafting a good rule to take the place of this ideal oracle. The most obvious rule, of fixing $\ell$ leaf nodes for just one of the options $\ell \in \{2, 4, 8, 16, 32, 64\}$, does not work. Addressing this open question is one area of research in our ongoing work on VPCs. This touches on an important topic in integer programming, of developing a better understanding of *cut selection*, especially considering the interaction between cutting and branching. The VPC framework provides an accessible way to explore this fertile area.

## 4 Conclusion

Our computational and theoretical results show that VPCs present a step forward in disjunctive cut generation. The cuts are strong relative to existing cuts, and they have the potential to significantly reduce branch-and-bound time. However, the question of how to realize that potential hinges on effectively selecting the disjunction size. It is promising that our results indicate that it may suffice to learn to select among only seven sizes. Moreover, the results are likely to improve with work on strengthening VPCs. Thus, we hope our encouraging early computational results lay a path forward to VPCs having practical impact in the near future.

## 5 Acknowledgements

Thank you to Egon Balas for his comments on this summary of our paper. Our research was supported by NSF grant DMI352885 and ONR contract N00014-03-1-0133.

## References

[1] T. Achterberg, T. Koch, and A. Martin. “MIPLIB 2003”. *Oper. Res. Lett.* 34.4 (2006), 361–372.

[2] E. Balas. “Disjunctive programming”. *Ann. Discrete Math.* 5 (1979), 3–51.

[3] E. Balas, S. Ceria, and G. Cornuéjols. “A lift-and-project cutting plane algorithm for mixed 0-1 programs”. *Math. Program.* 58.3, Ser. A (1993), 295–324.

[4] E. Balas and A. M. Kazachkov. “$\mathcal{V}$-polyhedral disjunctive cuts”. Working paper. Dec. 2018.

[5] D. E. Bell and M. L. Fisher. “Improved integer programming bounds using intersections of corner polyhedra”. *Math. Program.* 8 (1975), 345–368.

[6] R. E. Bixby, E. A. Boyd, and R. R. Indovina. “MIPLIB: a test set of mixed integer programming problems”. *SIAM News* 25 (2 1992), 16.

[7] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh. “An updated mixed integer programming library: MIPLIB 3.0”. *Optima* 58 (June 1998), 12–15.

[8] F. Cadoux and C. Lemaréchal. “Reflections on generating (disjunctive) cuts”. *EURO Journal on Computational Optimization* 1.1-2 (2013), 51–69.

[9] “Computational Optimization Research at Lehigh. MIP instances”. `http://coral.ise.lehigh.edu/data-sets/mixed-integer-instances/`. Accessed September 2017.

[10] R. E. Gomory. “Outline of an algorithm for integer solutions to linear programs”. *Bull. Amer. Math. Soc.* 64 (1958), 275–278.

[11] Gurobi Optimization, Inc. “Gurobi Optimizer Reference Manual”. `http://www.gurobi.com`. Version 7.5.1. 2017.

[12] A. M. Kazachkov. “Non-recursive cut generation”. Ph.D. Dissertation. Carnegie Mellon University, 2018.

[13] A. M. Kazachkov, S. Nadarajah, E. Balas, and F. Margot. “Partial hyperplane activation for generalized intersection cuts”. https://arxiv.org/abs/1703.02221. Working paper. 2017.

[14] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. “MIPLIB 2010: mixed integer programming library version 5”. *Math. Program. Comput.* 3.2 (2011), 103–163.

[15] R. Lougee-Heimer. “The Common Optimization INterface for Operations Research: promoting open-source software in the operations research community”. *IBM Journal of Research and Development* 47 (2003).

[16] Q. Louveaux, L. Poirrier, and D. Salvagnin. “The strength of multi-row models”. *Math. Program. Comput.* 7.2 (2015), 113–148.

[17] M. Miltenberger, T. Ralphs, and D. E. Steffy. “Exploring the numerics of branch-and-cut for mixed integer linear optimization”. *Operations Research Proceedings 2017*. Ed. by N. Kliewer, J. F. Ehmke, and R. Borndörfer. Cham: Springer International Publishing, 2018, 151–157.

[18] M. Perregaard and E. Balas. “Generating cuts from multiple-term disjunctions”. *Integer Programming and Combinatorial Optimization*. Vol. 2081. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, 348–360.