Computing the Spark of a Matrix

Andreas M. Tillmann
Operations Research Group & Visual Computing Institute
RWTH Aachen, Germany
Email: andreas.tillmann@cs.rwth-aachen.de

Marc E. Pfetsch
Research Group Optimization
TU Darmstadt, Germany
Email: pfetsch@mathematik.tu-darmstadt.de

***Abstract*—The spark of a matrix, i.e., the smallest number of linearly dependent columns, plays an important role in sparse signal recovery and compressed sensing; for instance, it yields characterizations of unique reconstructability. We develop several approaches to tackle the NP-hard problem of computing the spark; in particular, we propose a novel branch & cut scheme based on an integer program arising from a matroid circuit formulation. The potential advantage of our algorithm compared to using general-purpose solvers (applied to a mixed-integer programming model) is demonstrated in some numerical experiments.**

I. Introduction

The spark of a matrix is defined as

$$\operatorname{spark}(\boldsymbol{A}) \coloneqq \min\{\|\boldsymbol{x}\|_0 : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0},\, \boldsymbol{x} \neq \boldsymbol{0}\}.$$

In the context of compressed sensing and the sparse recovery problem

$$\min \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{P_0}$$

the spark is essential regarding uniqueness of solutions (e.g., every k-sparse vector $\hat{\boldsymbol{x}}$ uniquely solves (P_0) with $\boldsymbol{b} \coloneqq \boldsymbol{A}\hat{\boldsymbol{x}}$ if and only if $2k < \operatorname{spark}(\boldsymbol{A})$); it is also a key ingredient in identifying ambiguities in linear sensor arrays (see, e.g., [1]), and other applications.

The spark of a matrix $\boldsymbol{A}$ is also known as the *girth* of the matroid[1] associated with the columns of $\boldsymbol{A}$, sometimes called the linear or vector matroid induced by $\boldsymbol{A}$, for which the (in-)dependent sets are given by sets of linearly (in-)dependent columns. Spark computation was shown to be NP-hard in [2, Theorem 1 (cf. Corollary 1)]. Although, via matroid duality, one could in principle employ algorithms designed for computing the co-girth (see, e.g., [4], [5], [6]), no methods to directly determine $\operatorname{spark}(\boldsymbol{A})$ appear to be known so far except the obvious brute-force approach.

Throughout, we assume w.l.o.g. that $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{A}) = m < n$ and $\boldsymbol{A}_j \neq \boldsymbol{0}$ for all $j \in [n] \coloneqq \{1, \dots, n\}$, which is the natural situation (and can always be guaranteed by preprocessing).

II. Polynomial-Time Solvable Special Cases

Despite NP-hardness in general, the spark can be obtained efficiently for certain matrix classes. The following result generalizes well-known special cases such as $\boldsymbol{A}$ being the oriented incidence matrix of a graph G (representing the associated graphical matroid), for which $\operatorname{spark}(\boldsymbol{A})$ can be computed in polynomial time (poly-time, for short) as the length of the shortest cycle in G.

Theorem 1. *If $\boldsymbol{A}$ is unimodular (i.e., every $m \times m$ submatrix $\boldsymbol{A}'$ has $\det(\boldsymbol{A}') \in \{0, \pm 1\}$), then $\operatorname{spark}(\boldsymbol{A})$ can be determined in poly-time.*

Due to space limitations, we omit the proof; it uses the following auxiliary result, which may be of interest in its own right.

Proposition 2. *Problem (P_0) can be solved in poly-time (by means of the corresponding ℓ_1-minimization problem) if $\boldsymbol{b} \in \{0, \pm c\}^m$ for some $c \in \mathbb{R}$ and the matrix $\boldsymbol{A}$ is unimodular.*

M. E. Pfetsch is supported by the EXPRESS project within the SPP 1798 funded by the German Research Foundation (DFG).

[1] We refer to [3] for details on matroid theory.

III. (Mixed-) Integer Programming Approaches

To compute $\operatorname{spark}(\boldsymbol{A})$ for a given matrix $\boldsymbol{A}$, we may apply general-purpose mixed-integer program (MIP) solvers to the model

$$\min_{\boldsymbol{x}\in\mathbb{R}^n;\, \boldsymbol{y},\boldsymbol{z}\in\{0,1\}^n} \Big\{\mathbb{1}^\top \boldsymbol{y} \,:\, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0},\, -\boldsymbol{y}+2\boldsymbol{z} \leq \boldsymbol{x} \leq \boldsymbol{y},\, \mathbb{1}^\top \boldsymbol{z} = 1\Big\}. \tag{1}$$

Note that sign symmetry in nullspace vectors is eliminated by setting a specific x_j to 1 (the choice is left to the solver via the binary variables $\boldsymbol{z}$) and that $-\mathbb{1} \leq \boldsymbol{x} \leq \mathbb{1}$ holds implicitly.

To avoid the increase in variables and obtain a pure integer (binary) program, we can employ the following result on circuits in matroids, i.e., the inclusion-wise minimal dependent sets.

Lemma 3. *Let $\mathcal{M}$ be a matroid over E. Then, $C \subseteq E$ contains a circuit of $\mathcal{M}$ if and only if $C \cap \overline{B} \neq \emptyset$ for all bases B of $\mathcal{M}$.*

Since $\operatorname{spark}(\boldsymbol{A})$ is the length of a shortest circuit in the associated linear matroid $\mathcal{M}[\boldsymbol{A}]$, we can compute it with the IP

$$\min_{y\in\{0,1\}^n} \Big\{\mathbb{1}^\top \boldsymbol{y} \,:\, \sum_{j\notin B} y_j \geq 1 \;\forall B \subset [n] : \operatorname{rank}(\boldsymbol{A}_B) = |B| = m\Big\}. \tag{2}$$

However, since there may be exponentially many bases B for $\mathcal{M}[\boldsymbol{A}]$, we do not want to write down all these constraints explicitly; instead, we may generate them dynamically within a branch & bound procedure, to cut off current fractional linear programming relaxation solutions $\hat{\boldsymbol{y}}$ in the iterative model strengthening process. Indeed, a maximally violated such cut can be found efficiently:

Theorem 4. *Given $\hat{\boldsymbol{y}} \in [0,1]^n$, a basis B of $\mathcal{M}[\boldsymbol{A}]$ such that $\sum_{j\notin B} y_j < 1$ can be found (or proven not to exist) in poly-time.*

Proof: Due to the matroid structure, the greedy algorithm is optimal: We traverse the indices in decreasing order of their $\hat{\boldsymbol{y}}$-values and add an index to B (starting with the empty set) if the corresponding column set $\boldsymbol{A}_B$ remains linearly independent (which can be checked using Gaussian elimination), until $|B| = m$. ∎

We implemented the branch & cut algorithm in SCIP [7]. Besides the basic separation routine based on Theorem 4, the algorithm includes several other aspects that aid the solving process; for instance, current nodes can be pruned if any solution remaining in the branch corresponds to linearly independent column subsets, and the nonzero structure of $\boldsymbol{A}$ can be exploited for inferring local variable fixings. Space limitation disallows us going into more detail here.

Some preliminary numerical experiments are gathered in Table I.

IV. Conclusion

Our specialized branch & cut algorithm allows to compute the spark of a given matrix more efficiently than commercial MIP-solvers applied to black-box models. More rigorous numerical tests and tuning algorithmic parameters are still on the agenda, as is looking into solving (1) with the additional separation of covering-type inequalities from (2) (note that $\boldsymbol{y}$ is the same in both models).

TABLE I

EXPERIMENTAL RESULTS FOR SEVERAL DETERMINISTIC SENSING MATRICES $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ CONSTRUCTED WITH THE METHOD FROM [8]. ALL INSTANCES WERE SOLVED TO OPTIMALITY BY EITHER SOLVER EXCEPT TWO FOR WHICH GUROBI REACHED A 1 HOUR TIME LIMIT (LABELLED "TIMEOUT" IN THE RUNTIME COLUMN), WHERE THE RESPECTIVE OPTIMALITY GAPS WERE STILL AT 200% AND 100%, RESPECTIVELY. THE TESTS WERE CARRIED OUT SINGLE-THREAD ON A LINUX 64-BIT QUAD-CORE MACHINE (2.8 GHZ, 8 GB RAM) RUNNING SCIP 3.2.1 WITH CPLEX 12.6.1 AS LP SOLVER, AND GUROBI 6.5.0, RESP. THE COLUMN LABELED μ ADDITIONALLY GIVES THE RESP. MUTUAL COHERENCE, WHICH IMPLIES LOWER BOUNDS $1 + 1/\mu(\boldsymbol{A})$ ON THE SPARK OF $\boldsymbol{A}$.

				Br. & Cut (model (2))		Gurobi (model (1))	
m	n	μ	spark	time [s]	nodes	time [s]	nodes
24	50	$1/2$	4	0.18	49	0.43	2258
41	50	$1/3$	8	0.15	47	0.43	880
41	100	$1/3$	6	1.27	269	75.29	47119
59	100	$1/3$	8	1.05	226	49.47	16280
83	100	$1/3$	8	0.86	96	3.44	6696
52	200	$1/2$	4	4.73	221	18.13	26033
68	200	$1/2$	4	2.46	205	9.78	15525
120	200	$1/4$	15	1175.71	21809	*timeout*	(862592)
131	200	$1/3$	8	10.67	200	355.60	288186
84	300	$1/2$	4	52.27	449	32.98	26550
109	500	$1/3$	6	229.50	1891	*timeout*	(218376)

REFERENCES

[1] A. MANIKAS AND C. PROUKAKIS, *Modeling and Estimation of Ambiguities in Linear Arrays*, IEEE Transactions on Signal Processing, 46 (1998), pp. 2166–2179.

[2] A. M. TILLMANN AND M. E. PFETSCH, *The Computational Complexity of the Restricted Isometry Property, the Nullspace Property, and Related Concepts in Compressed Sensing*, IEEE Transactions on Information Theory, 60 (2014), pp. 1248–1259.

[3] J. G. OXLEY, *Matroid Theory*, Oxford Graduate Texts in Mathematics, Oxford University Press 1992.

[4] K. KIANFAR, A. POURHABIB, AND Y. DING, *An integer programming approach for analyzing the measurement redundancy in structured linear systems*, IEEE Transactions on Automation Science and Engineering, 8 (2011), pp. 447–450.

[5] J. D. ARELLANO AND I. V. HICKS, *Degree of Redundancy of Linear Systems using Implicit Set Covering*, IEEE Transactions on Automation Science and Engineering, 11 (2014), pp. 274–279.

[6] J. D. ARELLANO, *Algorithms to Find the Girth and Cogirth of a Linear Matroid*, PhD thesis, Rice University, 2014.

[7] G. GAMRATH AND T. FISCHER AND T. GALLY AND A. M. GLEIXNER AND G. HENDEL AND T. KOCH AND S. J. MAHER AND M. MILTENBERGER AND B. MÜLLER AND M. E. PFETSCH AND C. PUCHERT AND D. REHFELDT AND S. SCHENKER AND R. SCHWARZ AND F.SERRANO AND Y. SHINANO AND S. VIGERSKE AND D. WENINGER AND M. WINKLER AND J. T. WITT AND J. WITZIG, *The SCIP Optimization Suite 3.2*, ZIB Tech. Rep. 15-60, 2016. Code available at www.scip.zib.de.

[8] M. A. IWEN, *Simple deterministically constructible RIP matrices with sublinear fourier sampling requirements*, Proceedings of the 43rd Annual Conference on Information Sciences and Systems (CISS), John Hopkins University, IEEE, 2009, pp. 870–875.