EFFICIENT NON-CONVEX GRAPH CLUSTERING FOR BIG DATA

Naveed Naimipour and Mojtaba Soltanalian*

Department of Electrical and Computer Engineering
University of Illinois at Chicago
Chicago, Illinois

ABSTRACT

Big data analysis is a fundamental research topic with extensive technical obstacles yet to be overcome. Graph clustering has shown promise in addressing big data challenges by categorizing otherwise unlabeled data—thus giving them *meaning*. In this paper, we propose a set of non-convex programs, generally referred to as Hard and Soft Clustering programs, that rely on matrix factorization formulations for enhanced computational performance. Based on such formulations, we devise clustering algorithms that allow for large data analysis in a more efficient manner than traditional convex clustering techniques. Numerical results confirm the usefulness of the proposed algorithms for clustering purposes and reveal their potential for usage in big data applications.

Index Terms— Big Data, Non-Convex Methods, Graph Clustering, Soft/Hard Clustering, Matrix Factorization

1. INTRODUCTION

The collection of large data sets, also known as big data, has become a focal point of the current technological landscape. Big data analysis refers to obtaining valuable information from data sets which are immensely large, unorganized, and difficult to process using traditional data mining techniques, and has a wide range of applications including, e.g., inference, prediction, communications, and imaging [1]. In recent years, the question of how to efficiently produce such useful information has been explored. Graph clustering is a technique that has shown promise in structuring large volumes of data in a reasonable amount of time.

Clustering, in its most general form, is the process of identifying and categorizing large data from pre-existing unlabeled data. Its immediate applications range from social media sites (e.g. Facebook) to online shopping entities (e.g. Amazon) [2]. Despite the existence of numerous clustering methods, there is still no consensus regarding the most efficient algorithm. For instance, due to their simplicity, graph clustering techniques such as K-means and spectral clustering have been studied extensively [3]. However, such techniques have had numerous pitfalls (including a large computational cost) that have prevented them from effectively overcoming big data's biggest obstacles.

A major challenge currently associated with convex clustering algorithms is their poor ability to handle missing data [4]. Powerful clustering algorithms need to have the ability to work regardless of missing data, initializations, etc. To combat such issues, hierarchical clustering techniques have been devised to use non-convexity as a tool for data analysis. However, such methods are not yet capable of performing efficiently as abnormalities in data sets occur [5]. Moreover, techniques such as kernal-based methods have been able to deal with missing data points, but have not been able to do so consistently [6]. Alternatively, an approach that has also seen some success in clustering relates to matrix factorization. Specifically, low-dimensional representations of high dimensional data have been analyzed and have shown promise in terms of efficiency [7-9].

The purpose of this work is to approach graph clustering through a non-convex methodology stemming from a matrix factorization formulation. In particular, we propose two non-convex clustering formulations, referred to as *Hard Clustering* and *Soft Clustering* programs. Based on such formulations, we devise algorithms that allow the clusters of large data sets to be computed more efficiently and consistently when compared to traditional convex clustering techniques. We then extend our results by proposing another set of programs called *2-Hard Clustering* and *2-Soft Clustering* that are even more efficient than the original programs. This is further illustrated through numerical simulations performed to test the speed of each algorithm when compared to the improved ALM algorithm proposed in [10-11].

The rest of this paper is organized as follows: Section 2 introduces the problem formulation and foundations of our proposed clustering programs. This includes our proposed non-convex programs and details regarding their characteristics. Section 3 introduces the proposed algorithm derived from our programs and an even faster set of programs based on the original methods. Section 4 presents the numerical results, examining the efficiency of the proposed methods as compared to those in [10-11]. Finally, we conclude with a summary of the work and possible future research directions.

* Corresponding author (e-mail: *nnaimi2@uic.edu*). This work was supported in part by U.S. National Science Foundation Grant CCF-1704401.

2. PROBLEM FORMULATION

We begin with two definitions that distinguish between hard and soft clustering, and as a result, laying the foundation for most that follows.

Definition 1. *Let $B = \{0, 1\}$. We call $\mathbf{X} \in B^{n\times k}$ a k-clustering matrix if and only if each row of $\mathbf{X}$ has exactly one 1. The subset of k-clustering matrices of $B^{n\times k}$ will be denoted by $\mathcal{H}_{n,k}$.*

It should be noted that any $\mathbf{X} \in \mathcal{H}_{n,k}$ partitions the graph nodes to at most k clusters. The number of clusters is given by the number of nonzero columns of $\mathbf{X}$.

Definition 2. *We call $\mathbf{X} \in \mathbb{R}^{n\times k}$ a soft k-clustering matrix if and only if the elements of $\mathbf{X}$ are nonnegative and the sum of the entries at each row is equal to one. The associated subset of $\mathbb{R}^{n\times k}$ will be denoted by $\Omega_{n,k}$.*

The matrix $\mathbf{X} \in \Omega_{n,k}$ encodes the probability of any graph node to belong to any of the k clusters. Therefore, $\Omega_{n,k}$ is a more relaxed (or *softer*) version of $\mathcal{H}_{n,k}$ (as $\Omega_{n,k} \subset \mathcal{H}_{n,k}$) in which we are not forced to fully decide on the assignment of any node to any cluster. This may eliminate the false decision errors associated with $\mathcal{H}_{n,k}$.

2.1. Clustering Programs

Let $\mathbf{A}$ denote the adjacency matrix of the graph to be clustered. Based on the above definitions, we propose the following Hard and Soft Clustering programs,

(a) Hard Clustering:

$$\min_{\mathbf{X}\in\mathcal{H}_{n,k}} \|\mathbf{A} - \mathbf{X}\mathbf{X}^T\|_F, \tag{1}$$

(b) Soft Clustering:

$$\min_{\mathbf{X}\in\Omega_{n,k}} \|\mathbf{A} - \mathbf{X}\mathbf{X}^T\|_F. \tag{2}$$

Observe that the choice of $\mathbf{X}\mathbf{X}^T$ is not arbitrary:

1. For $\mathbf{X} \in \mathcal{H}_{n,k}$, the (l, m)-entry of $\mathbf{X}\mathbf{X}^T$ is one if and only if both the nodes l and m occur in the same cluster.
2. For $\mathbf{X} \in \Omega_{n,k}$, it can be easily verified that for any $l \neq m$, the (l, m)-entry of $\mathbf{X}\mathbf{X}^T$ represents the probability of the scenario in which both nodes occur in the same cluster.

Alternatively, the clustering task can be accomplished via the following over-parametrized *almost-equivalent* non-convex programs,

(a) Hard Clustering:

$$\min_{\mathbf{X}\in\mathcal{H}_{n,k},\mathbf{Q}\in\mathbb{R}^{n\times k}} \|\mathbf{X}\mathbf{Q} - \mathbf{A}^{1/2}\|_F \quad \text{s.t.} \quad \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_k \tag{3}$$

(b) Soft Clustering:

$$\min_{\mathbf{X}\in\Omega_{n,k},\mathbf{Q}\in\mathbb{R}^{n\times k}} \|\mathbf{X}\mathbf{Q} - \mathbf{A}^{1/2}\|_F \quad \text{s.t.} \quad \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_k \tag{4}$$

in which $\mathbf{Q}$ is a semi-unitary matrix since $\mathbf{Q}^T\mathbf{Q} \neq \mathbf{I}_n$. The above non-convex programs pave the way for an efficient computational approach to the clustering problem at hand. It is worth noting that the objectives in (1)-(2) become *small* (or zero) if and only if the objectives in (3)-(4) become *small* (or zero). Moreover, note that we can generally assume the diagonal entries of the adjacency matrix $\mathbf{A}$ are one or large. On the other hand, the diagonal entries of $\mathbf{X}\mathbf{X}^T$ in Soft Clustering represent the squared l_2-norms of rows of $\mathbf{X}$. Observe that for a vector with a fixed-sum of entries, *the sparser the vector, the larger its l_2-norm*. As a result, the minimization of the Soft Clustering objectives encourages a *sparse clustering*—i.e. the node association probabilities will be distributed as *sparsely* as possible, even without resorting to a sparsity constraint.

3. PROPOSED ALGORITHMS

In the following, we show that the non-convex optimization problems in (3)-(4) can be handled very efficiently using a cyclic approach. For given $\mathbf{X}$ (in $\mathcal{H}_{n,k}$ or $\Omega_{n,k}$) and $n \geq k$, we have:

$$\begin{aligned} &\|\mathbf{X}\mathbf{Q} - \mathbf{A}^{1/2}\|_F^2 \\ &= \text{tr}\left(\mathbf{A}^{T/2}\mathbf{A}^{1/2} + \mathbf{X}\mathbf{Q}\mathbf{Q}^T\mathbf{X}^T\right) - 2\text{Re}\{\text{tr}\{\mathbf{A}^{T/2}\mathbf{X}\mathbf{Q}\}\} \\ &= C - \|\mathbf{X}^T\mathbf{A}^{1/2} - \mathbf{Q}\|_F^2 \end{aligned} \tag{5}$$

where C is a constant. Now, let $\mathbf{X}^T\mathbf{A}^{1/2} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\mathbf{H}}$ represent the economy-size SVD of $\mathbf{X}^T\mathbf{A}^{1/2}$. Then, the minimizer semi-unitary matrix $\mathbf{Q}$ of (5) is obtained as [12]

$$\mathbf{Q}_* = \mathbf{U}\mathbf{V}^{\mathbf{H}}. \tag{6}$$

In addition, for fixed $\mathbf{Q}$, the optimal $\mathbf{X}$ of (3)-(4) is a nearest-matrix problem aiming to minimize $\|\mathbf{X} - \mathbf{A}^{1/2}\widehat{\mathbf{Q}}\|_F^2$, where $\widehat{\mathbf{Q}}$ is a unitary matrix formed by *completing* $\mathbf{Q}$ to serve as a basis for the whole $\mathbb{R}^n$. Note that:

1. If $\mathbf{X}_* \in \mathcal{H}_{n,k}$, then the location of the largest entry in any row of $\mathbf{A}^{1/2}\widehat{\mathbf{Q}}$ is the location of the single 1 entry in the corresponding row in $\mathbf{X}_*$.
2. If $\mathbf{X}_* \in \Omega_{n,k}$, then the optimal $\mathbf{X} = \mathbf{X}_*$ can be obtained in a row-wise manner (each independent from other rows). For a generic row of $\mathbf{A}^{1/2}\widehat{\mathbf{Q}}$, say $\boldsymbol{\alpha} \in \mathbb{R}^k$, we solve the following problem:

$$\begin{array}{ll} \min_{\mathbf{p}} & \|\mathbf{p} - \boldsymbol{\alpha}\|_2 \\ \text{s.t.} & p_l \geq 0\ ,\ l \in \{1, 2, ..., k\}, \\ & \sum_{l=1}^k p_l = 1, \end{array} \tag{7}$$

Take $\boldsymbol{\alpha}$ to Ω_k^+:

$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \left(\frac{1-\text{sum}(\boldsymbol{\alpha})}{k}\right)\mathbb{1}_k$.

If $\boldsymbol{\alpha}$ contains negative entries:

$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \min(\boldsymbol{\alpha})\mathbb{1}_k$.

Main Step:

Let $\bar{\boldsymbol{\alpha}}$ consist of the nonzero entries of $\boldsymbol{\alpha}$.

If $(1-\bar{\boldsymbol{\alpha}})/\text{length}(\bar{\boldsymbol{\alpha}}) < \min \bar{\boldsymbol{\alpha}}$:

$\bar{\boldsymbol{\alpha}} \leftarrow \bar{\boldsymbol{\alpha}} - \left(\frac{1-\text{sum}(\bar{\boldsymbol{\alpha}})}{\text{length}(\bar{\boldsymbol{\alpha}})}\right)\mathbb{1}_{\text{length}(\bar{\boldsymbol{\alpha}})}$.

[Done.]

Else,

$\bar{\boldsymbol{\alpha}} \leftarrow \bar{\boldsymbol{\alpha}} - \min(\bar{\boldsymbol{\alpha}})\mathbb{1}_{\text{length}(\bar{\boldsymbol{\alpha}})}$.

[Now we have even less nonzero values, and can proceed by using the Main Step on the newly obtained $\boldsymbol{\alpha}$.]

Algorithm 1: A recursive approach to find the closest probability vector to the given vector α.

which finds the closest probability vector to a given vector in $\mathbb{R}^k$.

Next, we show that (7) can be solved very efficiently using a recursive procedure. To formulate the procedure, we first define

$$\Omega_k = \{\alpha \in \mathbb{R}^k : \alpha_l \geq 0,\ \Sigma_{l=1}^k \alpha_l = 1\}, \tag{8}$$

$$\Omega_k^+ = \{\alpha \in \mathbb{R}^k : \alpha_l \geq 0,\ \Sigma_{l=1}^k \alpha_l \geq 1\}, \tag{9}$$

and resort to the following lemmas to cast the recursions.

Lemma 1. *Impact of identical additions:* $\arg\min_{\mathbf{p}\in\Omega_k} \|\mathbf{p}-\boldsymbol{\alpha}\|_2 = \arg\min_{\mathbf{p}\in\Omega_k} \|\mathbf{p}-(\boldsymbol{\alpha}+\lambda\mathbb{1}_k)\|_2$.

Lemma 2. *Suppose $\boldsymbol{\alpha} \in \Omega_k^+$ and*

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in\Omega_k} \|\mathbf{p}-\boldsymbol{\alpha}\|_2. \tag{10}$$

If $\alpha_l = 0$ for any $l \in \{1,2,...,k\}$ then $p_l^* = 0$.

Lemma 3. *If $\boldsymbol{\alpha} \in \Omega_k$ then $\boldsymbol{\alpha} = \arg\min_{\mathbf{p}\in\Omega_k} \|\mathbf{p}-\alpha\|_2$.*

The above claims are not difficult to verify and their proofs are thus omitted for the sake of brevity. Based on the above lemmas, we propose a set of steps, summarized in Algorithm 1, that obtain the vector that is the closest probability vector to the original α in $\mathbb{R}^k$. It is interesting to note that the probability vectors obtained by Algorithm 1 are sparse *by construction*, which is in agreement with our previous discussions on the Soft Clustering programs.

3.1. Further Enhancement of Computational Efficiency

Further analysis of the Hard and Soft Clustering programs indicates even higher efficiency with the use of $\mathbf{A}$ instead of $\mathbf{A}^{1/2}$ in the formulations. Observe that if $\mathbf{A}$ is an adjacency matrix that encodes information on paths of length 1, $\mathbf{A}^2$ reveals relevant information on paths of length 2. However, for clustering purposes, we can assume that each cluster forms a complete sub-graph. Hence, we can perform the clustering based on the paths of length 2 that eliminate the need for computing the square-root of $\mathbf{A}$. More precisely, we can consider the programs:

(a) 2-Hard Clustering:

$$\min_{\mathbf{X}\in\mathcal{H}_{n,k},\mathbf{Q}\in\mathbb{C}^{n\times k}} \|\mathbf{XQ}-\mathbf{A}\|_F \text{ s.t. } \mathbf{QQ}^T = \mathbf{I}_k, \tag{11}$$

(b) 2-Soft Clustering:

$$\min_{\mathbf{X}\in\Omega_{n,k},\mathbf{Q}\in\mathbb{C}^{n\times k}} \|\mathbf{XQ}-\mathbf{A}\|_F \text{ s.t. } \mathbf{QQ}^T = \mathbf{I}_k. \tag{12}$$

As a final remark, the benefits of our methods in adaptive scenario where we deal with an updated $\mathbf{A}$ should not be overlooked. The proposed method will be very efficient in such cases as minor modifications in $\mathbf{A}$ typically result in minor modifications in the clustering outcome. Therefore, a fast convergence will be achieved compared to methods that require full analysis of $\mathbf{A}$, or finding the clusters from scratch.

4. NUMERICAL RESULTS

We begin the numerical results by investigating the ability of the proposed methods to produce the correct clusters for different sizes of a (possibly noisy) adjacency matrix. Several clustering examples are shown in Figs. 1-2. In particular, we observed that Hard Clustering performs very well when the noise power is low. On the other hand, Soft Clustering is generally more robust to strong noise. Furthermore, 2-Soft Clustering was observed to be *harder* than Soft Clustering. Specifically, 2-Soft Clustering typically obtained the exact clustering solution and worked better than 2-Hard Clustering.

With this general behavior in mind, we present hereafter the numerical results assessing the speed of the proposed algorithms compared to the improved ALM algorithm [10-11]. In each case, the completion time of the algorithms was plotted with varying n while k was kept constant.

Fig. 3 shows the results of our analysis on Hard and Soft Clustering for large matrices where the value of k is set as 8 and 16, respectively. It can be seen that Hard and Soft Clustering consistently outperform the improved ALM algorithm in terms of computational cost. As the value of n increases, our programs perform even better relative to the improved ALM

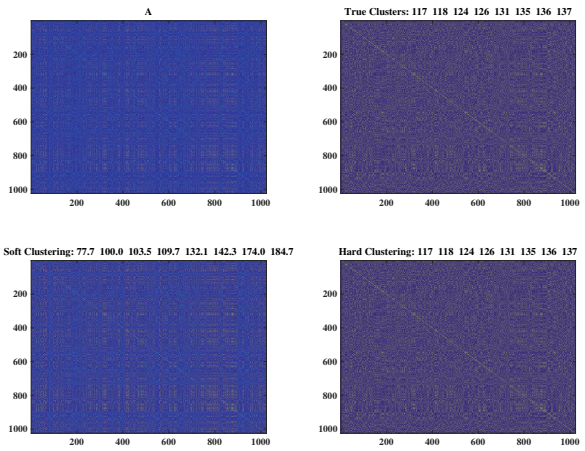

Fig. 1: Graph clustering completion via the Hard and Soft Clustering formulations ($n = 1024$, $k = 8$). Hard Clustering finds the clusters correctly in a low-noise scenario.

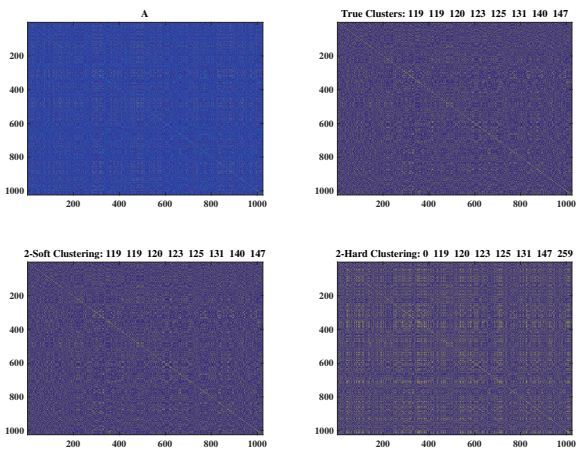

Fig. 2: Graph clustering completion via the 2-Hard and 2-Soft Clustering formulations ($n = 1024$, $k = 8$). The 2-Soft Clustering approach correctly identifies the clusters.

algorithm, which shows their potential for big data. As expected, the proposed Soft Clustering algorithm appears to be slightly more costly than its Hard Clustering counterpart.

A similar numerical analysis was performed for the 2-Hard and 2-Soft programs. It can be seen that both 2-Hard and 2-Soft Clustering programs consistently outperform the improved ALM algorithm by an even wider margin than our original clustering programs. Once again, the potential for applications in big data scenarios is evident.

5. CONCLUSION

Efficient processing and analysis of large data sets using clustering algorithms is an ongoing research topic. Inspired by matrix factorization formulations, we presented a set of non-convex Hard and Soft Clustering programs (and their associated algorithms) that have shown great potential for big data analysis. We further extended our formulations and introduced another set of clustering programs called 2-Hard and 2-Soft Clustering, that were even faster than the original set of suggested programs. The proposed methods were shown to outperform the improved ALM algorithm by a wide margin in terms of computational efficiency.

6. ACKNOWLEDGEMENT

The authors would like to thank Ramya Korlakai Vinayak and Prof. Babak Hassibi from California Institute of Technology for providing us with the MATLAB code for the improved ALM algorithm, as well as useful discussions that helped us significantly in clarifying the ideas of this work.

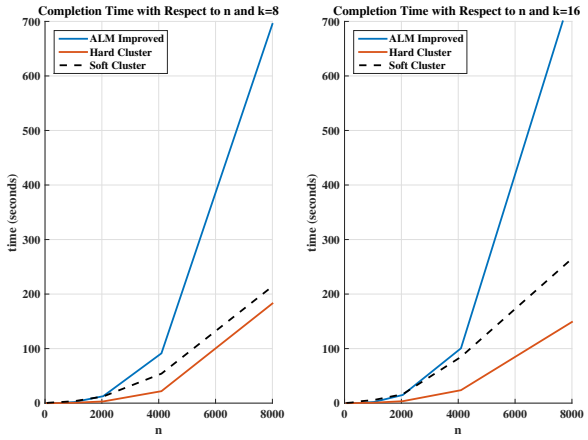

Fig. 3: Completion time of Hard and Soft Clustering methods with $k \in \{8, 16\}$.

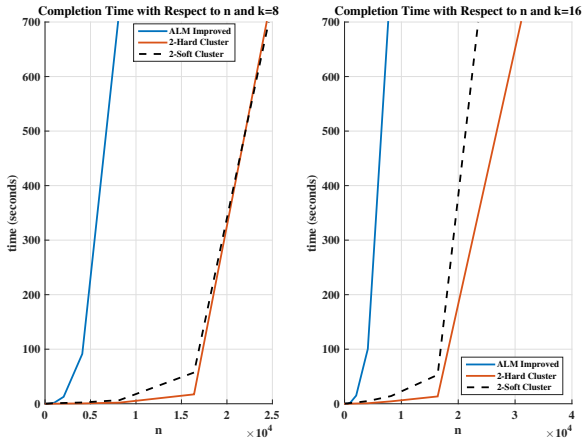

Fig. 4: Completion time of 2-Hard and 2-Soft Clustering methods with $k \in \{8, 16\}$.

7. REFERENCES

[1] P. Zikopoulos, D. deRoos, and K. P. Corrigan, *Harness the Power of Big Data*. New York: McGraw-Hill, 2012.

[2] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert Tarjan. "Clustering Social Networks," *Algorithms and Models for the Web-Graph, volume 4863 of Lecture Notes in Computer Science*, chapter 5, pages 56-67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[3] C. Boutsidis, A. Zouzias, M. Mahoney, and P. Drineas, "Randomized dimensionality reduction for K-means clustering," *Comput. Res. Repos.*, 2011.

[4] Yudong Chen, Sujay Sanghavi, and Huan Xu. "Clustering sparse graphs." *NIPS*, pages 2213-2221, 2012.

[5] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236-244, 1963.

[6] R. J. Hathaway and J. C. Bezdek, "Fuzzy c-means clustering of incomplete data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 5, pp. 735-744, 2001.

[7] Venkat Chandrasekaran, Sujay Sanghavi, Pablo A. Parrilo, and Alan S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572-596, 2011.

[8] L. De Lathauwer and J. Castaing, "Blind identification of underdetermined mixtures by simultaneous matrix diagonalization," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1096-1105, 2008.

[9] J. F. Tenenbaum, V. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319-2323, 2000.

[10] R. K. Vinayak, S. Oymak, and B. Hassibi, "Graph clustering with missing data: Convex algorithms and analysis," in *Proc. 27th Adv. Neural Inf. Process. Syst.*, 2014, pp. 2996-3004.

[11] R. K. Vinayak, S. Oymak and B. Hassibi, "Sharp performance bounds for graph clustering via convex optimization," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, 2014, pp. 8297-8301.

[12] P. Stoica, J. Li, X. Zhu, "Waveform synthesis for diversity-based transmit beampattern design," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2593-2598, 2008.