STRONGER RECOVERY GUARANTEES FOR SPARSE SIGNALS EXPLOITING COHERENCE STRUCTURE IN DICTIONARIES

Eeshan Malhotra, Ajit Rajwade

Indian Institute of Technology Bombay
Department of Computer Science and Engineering

Karthik S. Gurumoorthy

International Center for Theoretical Sciences
Tata Institute of Fundamental Research

ABSTRACT

This paper presents a method for improving the recovery guarantee for signals that are sparse or compressible in some general basis (dictionary) using a splitting and reordering approach. The splitting algorithm applies existing results for dictionaries that are naturally characterized as a concatenation of two sub-parts, to arbitrary dictionaries, by devising the optimal artificially induced split in the dictionary. A complete approach is presented for partitioning arbitrary dictionaries into two parts, so as to obtain the optimal coherence bounds on recovery, along with a proof of optimality. A heuristic is provided for recursive application of the splitting algorithm to further improve upon these bounds, using a multi-way dictionary split. We analyze cases where an appropriate split in the dictionary predicts less conservative signal sparsity bounds for successful recovery than those considering the dictionary as a monolithic block. Our present work does not provide a new algorithm for sparse signal recovery but rather mines for structures in the dictionary, towards strengthening the existing coherence-based recovery bounds.

Index Terms— Sparse signal recovery, coherence bound, recovery guarantee, dictionary splitting

1. BACKGROUND AND CONTRIBUTION

Consider the setting familiar in compressive sensing ([1], [2]). We have measurements $\boldsymbol{y} \in \mathbb{R}^M$ from the k-sparse signal $\boldsymbol{x} \in \mathbb{R}^N$ generated using the model

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x} + \boldsymbol{\eta} \tag{1}$$

where $\boldsymbol{D} \in \mathbb{R}^{M \times N}$ is the measurement matrix or dictionary with unit-normalized columns, and $\boldsymbol{\eta} \in \mathbb{R}^M$ is the measurement noise, with $||\boldsymbol{\eta}||_2 \leq \epsilon$.

The signal $\boldsymbol{x}$ can be recovered using basis pursuit ([3]), and a coherence-based recovery guarantee is well known in compressive sensing literature ([4]). In special cases, the bound can be improved.

Thanks to support from IIT Bombay Seed Grant #14IRCCSG012

This work also benefited from the support of the AIRBUS Group Corporate Foundation Chair in Mathematics of Complex Systems established in ICTS-TIFR.

Existing work ([5], [6], [7]) explores the case where $\boldsymbol{D}$ and $\boldsymbol{x}$ have the special structure:

$$\boldsymbol{D} = [\boldsymbol{A}\ \boldsymbol{B}]\ ,\ \ \boldsymbol{x}^T = [\boldsymbol{x}_a^T\ \boldsymbol{x}_b^T]$$

where $\boldsymbol{A} \in \mathbb{R}^{M \times N_a}$, $\boldsymbol{B} \in \mathbb{R}^{M \times N_b}$, $\boldsymbol{x}_a \in \mathbb{R}_a^N$, $\boldsymbol{x}_b \in \mathbb{R}_b^N$, $N_a + N_b = N$. We specifically use results from [7], which show that the guarantee can be expressed in terms of the coherence values of the individual sub-dictionaries $\boldsymbol{A}$ and $\boldsymbol{B}$.

$$\mu_a := \max_{i,j,i \neq j} |\boldsymbol{A}_i^T \boldsymbol{A}_j|, \qquad \mu_b := \max_{i,j,i \neq j} |\boldsymbol{B}_i^T \boldsymbol{B}_j|,$$

and their mutual coherence, defined as

$$\mu_m := \max_{i,j} |\boldsymbol{A}_i^T \boldsymbol{B}_j|.$$

Consequently, the coherence for dictionary $\boldsymbol{D}$ can be expressed as

$$\mu_d := \max_{i,j,i \neq j} |\boldsymbol{D}_i^T \boldsymbol{D}_j| = \max\{\mu_a, \mu_b, \mu_m\}$$

where $\boldsymbol{X}_i$ indicates the i^{th} column of matrix $\boldsymbol{X}$.
Assuming without loss of generality that $\mu_b \leq \mu_a$, the basis pursuit bound, then, says (Theorem 4 in [7]):
If

$$k < \max\left\{ \frac{2(1+\mu_a)}{\mu_a + 2\mu_d + \sqrt{\mu_a^2 + \mu_m^2}}, \frac{1+\mu_d}{2\mu_d} \right\} \tag{2}$$

then the solution $\hat{\boldsymbol{x}}$ of the basis pursuit problem

$$\begin{aligned} &\text{minimize } ||\tilde{\boldsymbol{x}}||_1 \\ &\text{subject to } ||\boldsymbol{y} - \boldsymbol{D}\tilde{\boldsymbol{x}}||_2 \leq \eta \end{aligned}$$

$$\text{satisfies } ||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2 \leq C(\epsilon + \eta) + C'(||\boldsymbol{x} - \boldsymbol{x}_s||_1) \tag{3}$$

where k is the number of non-zero entries in $\boldsymbol{x}$, $\boldsymbol{x}_s$ is $\boldsymbol{x}$ after setting all but the k largest components to 0, and $C, C' \geq 0$ are constants. Further, it holds that (Appendix D, [7])

$$(1 - \hat{\delta})||\boldsymbol{x}||_2^2 \leq ||\boldsymbol{D}\boldsymbol{x}||_2^2 \leq (1 + \hat{\delta})||\boldsymbol{x}||_2^2 \tag{4}$$

with

$$\hat{\delta} = \min \left\{ \frac{1}{2} \left(\mu_a \left(k-2 \right) + k\sqrt{\mu_a^2 + \mu_m^2} \right),\ \mu_d(k-1) \right\} \tag{5}$$

The requirement in equation (2) improves on the bound for a general $\boldsymbol{D}$ (requiring $k \leq \frac{1+\mu_d}{2\mu_d}$).

This is the key result that we build on and extend. While the work of Studer and Baraniuk [7] applies to dictionaries and signals with an inherent characteristic structure composed of two well-separated parts, we apply the result to an arbitrary dictionary $\boldsymbol{D}$, by inducing a partition that optimizes the requirement on the sparsity of the signal. While the recovery algorithm itself is unchanged, our proposed algorithm produces a tighter coherence-based bound for recovery using basis pursuit, for a given dictionary $\boldsymbol{D}$.

2. TWO-WAY DICTIONARY SPLITTING

First, it is abundantly clear that swapping any two columns $\boldsymbol{D}_i$ and $\boldsymbol{D}_j$ of a dictionary, while swapping the corresponding elements $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ has no impact on the product $\boldsymbol{Dx}$. In fact, if $\boldsymbol{D'}$ is a permutation of the columns of $\boldsymbol{D}$, and $\boldsymbol{x'}$ the corresponding transformation of $\boldsymbol{x}$, then $\boldsymbol{Dx} = \boldsymbol{D'x'}$. Moreover, $\boldsymbol{x}$ and $\boldsymbol{x'}$ clearly have the same sparsity, and if the order of permutation is known, $\boldsymbol{x'}$ determines $\boldsymbol{x}$ exactly. Therefore, the problem of recovering the signal $\boldsymbol{x}$, given $\boldsymbol{D}$, and measurements $\boldsymbol{y} = \boldsymbol{Dx} + \boldsymbol{\eta}$ is equivalent to recovering $\boldsymbol{x'}$, given $\boldsymbol{D}$, $\boldsymbol{D'}$ and $\boldsymbol{y}$.

Secondly, for a given $\boldsymbol{D}$, the quantity in the RHS of equation (2) can be maximised directly by maximising the quantity

$$F = \frac{2(1+\mu_a)}{\mu_a + 2\mu_d + \sqrt{\mu_a^2 + \mu_m^2}} \tag{6}$$

where $\mu_a \geq \mu_b$.

We can now define the problem of dictionary split induction as: Given a dictionary $\boldsymbol{D} \in \mathbb{R}^{M \times N}$, devise matrices $\boldsymbol{A}$, $\boldsymbol{B}$, such that the columns of $\boldsymbol{D'}$ can be permuted to give $\boldsymbol{D}$, where $\boldsymbol{D'} = [\boldsymbol{A}\ \boldsymbol{B}]$, and F is maximised.

2.1. Algorithm

We solve the problem of dictionary splitting by mapping it to an equivalent graph problem. The transformation is as follows:

Let $G = (V, E)$ be a complete, undirected, weighted graph, with vertices V and edges E, such that each vertex corresponds to a column of the matrix $\boldsymbol{D}$, and the weight of edge uv equals the dot product of the columns corresponding to vertices u and v. The problem, then, is to define a cut through G that partitions it into two disjoint components, $A(V_a, E_a)$ and $B(V_b, E_b)$. Let E_x be all the edges crossing the cut, and V_x be the set of vertices connected by edges in E_x. Note that

(a) $V_a \subseteq V, V_b \subseteq V, V_a \cup V_b = V$

(b) $E_a \subseteq E, E_b \subseteq E, E_x \subseteq E, E_a \cup E_b \cup E_x = E.$

Let us also define the following quantities on the graph G:

$$\mu_a := \max_{e \in E_a} w(e)\ \ ;\ \ \mu_b := \max_{e \in E_b} w(e)$$
$$\mu_m := \max_{e \in E_x} w(e)$$
$$\mu_d := \max_{e \in E} w(e) = \max\left\{\mu_a, \mu_b, \mu_m\right\}$$

where $w(e)$ denotes the weight of edge e.

There is a one-to-one correspondence between the terms μ_a, μ_b, μ_m, μ_d defined on the graph G and the equivalent coherence values defined on the dictionary $\boldsymbol{D}$. The matrix $\boldsymbol{A}$ can be constructed by selecting the columns from $\boldsymbol{D}$ corresponding to sets V_a and concatenating them in an arbitrary order. Similarly, the matrix $\boldsymbol{B}$ can be constructed from V_b.

This is essentially a greedy algorithm. Intuitively, we try to process edges starting from the heaviest, and to add each to the set E_x of cut edges, as long as doing so does not create a contradiction in the colouring of vertices. After each step, the edges in consideration are those that are adjacent to edges already included in E_x. In the following iteration, the heaviest of these is selected as the candidate for inclusion.

Initially, each vertex is unassigned (i.e. $\mathit{colour}(v) = 0$), and the Q, the queue of edges yet to be processed, is empty ($Q = \phi$). As vertices are processed, they are assigned to set $\mathbf{A}(\mathit{colour}(v) = 1)$ or $\mathbf{B}(\mathit{colour}(v) = 2)$, and the edges emanating from the vertex are added to the queue Q.

This algorithm provides us with a two-way split in the dictionary that allows for the tightest recovery guarantee. The process for recovering the signal (i.e. basis pursuit) is unaltered.

2.2. Running time complexity

Let the number of vertices in graph G be n (This corresponds to the number of columns in dictionary $\boldsymbol{D}$). Steps 1 through 7 involve n sorting operations, each consuming time $\mathcal{O}(n \log n)$, for a total of $\mathcal{O}(n^2 \log n)$ operations. Steps 8 through 12 consist of one time assignments, and hence, can be performed in $\mathcal{O}(1)$ time.

The while loop in steps 13 through 23 runs for at most $\mathcal{O}(n^2)$ iterations, since each iteration processes one edge. All operations inside the loop are $\mathcal{O}(1)$ complexity, except the merge operation (step 18) and the pop operation (step 22). Although, with a naïve implementation, a merge might take time $\mathcal{O}(n^2)$, with the use of an appropriate data structure, such as a Fibonacci heap ([8]) for representing Q and L, we can get the amortised time complexity of a single merge operation down to $\mathcal{O}(1)$, while still getting a time complexity of a single pop operation of $\mathcal{O}(\log n^2) = \mathcal{O}(\log n)$. Since a merge operation only happens when we encounter an unprocessed vertex (inside the *if* clause), there are $\mathcal{O}(n)$ merges. A pop operation

Algorithm 1 Two-way Splitting Algorithm

for $v \in V$ **do**
  colour$(v) \leftarrow 0$
  $L(v) \leftarrow$ edges from v, in decreasing order of weight
end for
$E_x =$ cut edges $\leftarrow \phi$
Q = list of edges to process $\leftarrow \phi$
processed $\leftarrow \phi$

$e = uv \leftarrow$ heaviest edge in G
colour$(u) \leftarrow 1$
$L(u) \leftarrow L(u) \setminus \{e\}$
$Q \leftarrow L(u)$
processed $\leftarrow \{u\}$

while size(*processed*) $< n$ **do**
  if one of $\{u, v\}$ (say v) is s.t. *colour*$(v) == 0$ **then**
    if *colour*$(u) == 1$ **then** *colour*$(v) \leftarrow 2$
    if *colour*$(u) == 2$ **then** *colour*$(v) \leftarrow 1$
    $L(v) \leftarrow L(v) \setminus \{e\}$
    Merge $L(v)$ into Q
    processed $\leftarrow$ *processed* $\cup \{v\}$
  end if
  if *colour*$(u)!=$ *colour*(v) **then** $E_x \leftarrow E_x \cup \{e\}$
  $e = uv \leftarrow$ Pop heaviest edge from Q
end while

$\mathbf{V}_a \leftarrow \{v \mid$ *colour*$(v) == 1\}$
$\mathbf{V}_b \leftarrow \{v \mid$ *colour*$(v) == 2\}$
Output $\mathbf{V}_a$, $\mathbf{V}_b$

occurs in every iteration of the loop, so there are $\mathcal{O}(n^2)$ pops. Thus, all iterations collectively consume time $\mathcal{O}(n^2 \log n)$. Hence, overall, the algorithm consumes time $\mathcal{O}(n^2 \log n)$.

2.3. Proof of Optimality

Consider the heaviest edge in G, say $e^* = uv$. At the end of any possible assignment, either $\max\{\mu_a, \mu_b\} = w(e^*)$, or $\mu_m = w(e^*)$. With this consideration in mind, it is clear from equation (6) that the latter case is always advantageous i.e. to maximize F, $e^* \in E_x$. By the same logic, the same reasoning can be applied to the next heaviest edge, the ones after, and so on, until it is impossible to add an edge to E_x - because doing so will create an inconsistent assignment of the vertices. In other words, adding edge e to E_x will create an odd length cycle, which cannot possibly occur in a graph cut.

But this addition is precisely the kind that is not allowed by our algorithm. $E_x \leftarrow E_x \cup \{e\}$ only if one of the vertices is unassigned (*colour*$(v) == 0$). It can also be verified by an exchange argument that the edge e that is not included in E_x must be the lightest edge in the cycle.

Thus, the assignment returned by Algorithm 1 must be optimal i.e., the tightest bound can be provided by exploiting the dictionary structure as suggested by Algorithm 1.

3. HIGHER ORDER RECURSIVE ALGORITHM

Algorithm 1 gives us an optimal two-way split for $\boldsymbol{D}$. The split is optimal in the sense of maximising F in equation (6), and minimizing $\hat{\delta}$ in equation (5). The process, in theory, can be generalized to splitting $\boldsymbol{D}$ into any number of splits to improve the bound. However, to consider each possible split naively would require exponential time, and would be prohibitively expensive, for any reasonable size dictionary. We now describe a recursive, polynomial-time heuristic to create a multi-way split in $\boldsymbol{D}$, to further tighten the bounds.

Consider equation (5). When the value of $\hat{\delta}$ is equal to the first term in the minimization, we can imagine the dictionary $\boldsymbol{D}$ being equivalent to a hypothetical dictionary $\tilde{\boldsymbol{D}}$, with

$$\tilde{\mu_d}(k-1) = \frac{1}{2}\left(\mu_a\,(k-2) + k\sqrt{\mu_a^2 + \mu_m^2}\right). \qquad (7)$$

For this bound to be better than that obtained by considering no structure in the dictionary, $\tilde{\mu_d}$ must be less than the second term in the minimization in equation (5). That is, $\tilde{\mu_d} < \mu_d$. By construction, Algorithm 1 produces $\mu_m = \mu_d$ (Since the heaviest edge in the graph is first placed in the set of cut edges). Therefore, this condition can be expressed as

$$\frac{1}{2}\left(\mu_a\,(k-2) + k\sqrt{\mu_a^2 + \mu_m^2}\right) < \mu_m(k-1)$$

$$\implies \frac{\mu_a}{\mu_m} < 1 - \frac{k}{2}\left(1 - \sqrt{\frac{k-2}{k-1}}\right). \qquad (8)$$

As k increases, $\frac{\mu_a}{\mu_m}$ asymptotically converges to a value of $\frac{3}{4}$, as show in the graph below.

In fact, for large k, equation 7 reduces to:

$$\tilde{\mu_d} = \frac{1}{2}\left(\mu_a + \sqrt{\mu_a^2 + \mu_m^2}\right). \qquad (9)$$

That is, the role of coherence of the hypothetical dictionary is played by the quantity $\frac{1}{2}\left(\mu_a + \sqrt{\mu_a^2 + \mu_m^2}\right)$.

It also follows that $\tilde{\mu_d} \leq \mu_m = \mu_d$ when $\mu_a \leq \frac{3}{4}\mu_m$ (for large k), resulting in tighter bounds for recovery.

These results use the coherence values of sub-dictionaries **A** and **B** as μ_a, and μ_b respectively. However, we can recursively apply the dictionary splitting algorithm to each of these, to improve the *effective* coherence values ($\tilde{\mu_d}$ in equation 9), $\tilde{\mu_a}$, and $\tilde{\mu_b}$.

3.1. Algorithm

The algorithm presented below uses a greedy approach, using the optimal 2-way split at each level, leveraging equation

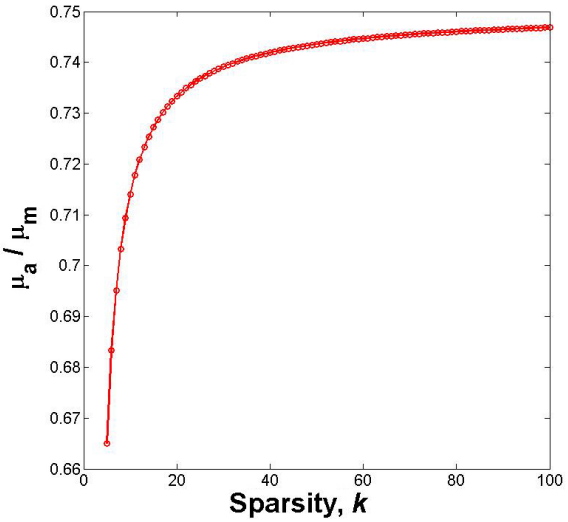

Fig. 1. μ_a/μ_m vs k

9 at each step to determine the equivalent coherence of the split pair. The algorithm uses a call to **Split_2Way**$(\boldsymbol{D})$, a function designed as presented in Algorithm 1, to return the optimal two-way split for dictionary $\boldsymbol{D}$. The multi-way split algorithm executed to depth l, achieves a 2^l-way split.

Algorithm 2 Depth l multi-way Split Algorithm

function EFFECTIVE_COHERENCE($\boldsymbol{D}, l$)
  $\boldsymbol{A}, \boldsymbol{B}$ = **Split_2Way**($\boldsymbol{D}$)
  $\mu_a \leftarrow$ **Effective_Coherence** $(\boldsymbol{A}, (l-1))$
  $\mu_b \leftarrow$ **Effective_Coherence** $(\boldsymbol{B}, (l-1))$
  $\mu_m \leftarrow \max_{i,j} |\boldsymbol{A}_i^T \boldsymbol{B}_j|$
  $\mu_d \leftarrow \max\{\mu_a, \mu_b, \mu_m\}$
  if l==1 **then**
    return μ_d
  else
    return $\min\left\{\mu_d, \frac{1}{2}\left(\mu_a + \sqrt{\mu_a^2 + \mu_m^2}\right)\right\}$
end function

$l \leftarrow$ maximum depth to explore
Output **Effective_Coherence** $(\boldsymbol{D}, l)$

Unfortunately, unlike Algorithm 1, this algorithm is not guaranteed to produce the optimal split from the set of all possible 2^l-way splits, and counter-examples can be constructed to prove its sub-optimality. However, it is guaranteed to produce an effective-coherence that is at least as good as the split in Algorithm 1, since this case is subsumed in Algorithm 2.

4. EMPIRICAL RESULTS

For empirical tests, each dictionary, $\boldsymbol{D}$, was designed by shuffling together the columns of two orthogonal matrices. $\boldsymbol{D}$ will have a low value of coherence. However, using the two-way splitting algorithm described in Section 2, we were able to reduce the effective coherence even further - perfectly splitting the two orthogonal matrices each time. The improvement in the bound on k and $\hat{\delta}$ vary with the size of the matrix. The details are summarized in the following table.

Unsplit dictionary			
N	200	500	1000
μ_d	0.141	0.0894	0.0632
Upper bound on k	4.036	6.090	8.406
$\hat{\delta}$	6.930	31.216	44.209
Split dictionary			
μ_a	0	0	0
μ_b	0	0	0
μ_m	0.141	0.0894	0.0632
μ_d	0.141	0.0894	0.0632
Effective coherence $\tilde{\mu_d}$	0.720	0.0448	0.0317
Upper bound on k	7.521	11.648	16.289
$\hat{\delta}$	3.536	15.652	22.136

For each experiment, the orthogonal matrices were perfectly recovered, giving us μ_a and μ_b of 0. $\mu_m = \mu_d$, by construction, as specified in Algorithm 1. The bound on k and $\hat{\delta}$ improved, typically by a factor of approximately 2 in each case.

Since our proposed method only presents an approach to improve the recovery bounds (like in Theorem 4, [7]), and not a new algorithm for actual recovery, we refrain from showcasing experiments that empirically compare the recovered signal to the true signal. We employ the well known basis pursuit denoising algorithms for recovery, and the retrieved signal will be *identical* to the scenario where our dictionary splitting technique is not used to calculate the bounds.

5. CONCLUSIONS

We extended existing results on recovery guarantees for sparse and compressible signals, using the fact that measurement matrices often incorporate some structure that may not be automatically visible. In fact, the first algorithm proposed in the paper discovers (in polynomial time) the optimal partition of the measurement matrix that provides the tightest bound with a two-part dictionary. Additionally, a further extension - the multi-way splitting algorithm is also presented, which provides a heuristic to extend the result by recursively applying the two-way splitting algorithm to each partition, and provides an even stronger bound on the recovery. While the multi-way splitting algorithm improves the bound, it is not guaranteed to return the optimal split (unlike the two-way splitting algorithm) - since the possible number of splits grows exponentially large.

Neither the algorithm for actually recovering the signal nor the objective function, namely $\min ||\boldsymbol{x}||_1$ s.t $||\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}||_2 <= \eta$, is modified. The splitting schemes proposed

only improve the theoretical bound on recovery for a given dictionary. Both algorithms are applicable to any generic dictionary. We hope to, in the future, provide an efficient algorithm for a general order, multi-way split in dictionaries as well.

6. REFERENCES

[1] Emmanuel J Candès and Michael B Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[2] Emmanuel J Candès, Justin Romberg, and Terence Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[3] Scott Shaobing Chen, David L Donoho, and Michael A Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[4] Tony Cai, Lie Wang, and Xu Guangwu, "Stable recovery of sparse signals and an oracle inequality," *IEEE Transactions on Information Theory*, vol. 7, no. 56, pp. 3516–3522, 2010.

[5] Patrick Kuppinger, Giuseppe Durisi, and Helmut Bolcskei, "Uncertainty relations and sparse signal recovery for pairs of general signal sets," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 263–277, 2012.

[6] Christoph Studer, Patrick Kuppinger, Graeme Pope, and Helmut Bolcskei, "Recovery of sparsely corrupted signals," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3115–3130, 2012.

[7] Christoph Studer and Richard G Baraniuk, "Stable restoration and separation of approximately sparse signals," *Applied and Computational Harmonic Analysis*, vol. 37, no. 1, pp. 12–35, 2014.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, MIT press, 2009.