Math. Program., Ser. B (2007) 109:263–281
DOI 10.1007/s10107-006-0024-1

FULL LENGTH PAPER

# Solving maximum-entropy sampling problems using factored masks

**Samuel Burer · Jon Lee**

Received: 28 February 2005 / Accepted: 26 October 2005 /
Published online: 19 September 2006


**Abstract** We present a practical approach to Anstreicher and Lee's masked spectral bound for maximum-entropy sampling, and we describe favorable results that we have obtained with a Branch-and-Bound algorithm based on our approach. By representing masks in factored form, we are able to easily satisfy a semidefiniteness constraint. Moreover, this representation allows us to restrict the rank of the mask as a means for attempting to practically incorporate second-order information.

**Keywords** Entropy · Branch and bound · Nonlinear programming · Eigenvalue

**Mathematics Subject Classification (2000)** 90C26 · 90C22 · 90C27 · 90C57

## 1 Introduction

Let $n$ be a natural number, and let $N := \{1, 2, \ldots, n\}$. Let $C$ be an order-$n$ symmetric positive definite matrix. Let $s$ be an integer between 0 and $n$. For subsets $S$ and $T$ of $N$, let $C[S, T]$ denote the submatrix of $C$ having rows indexed by $S$ and columns indexed by $T$. The *maximum-entropy sampling problem* is to calculate

Supported in part by NSF Grant CCR-0203426.

S. Burer
Department of Management Sciences, University of Iowa, Iowa City, IA 52242, USA
e-mail: samuel-burer@uiowa.edu

J. Lee (✉)
Department of Mathematical Sciences, T.J. Watson Research Center, IBM,
Yorktown Heights, NY 10598, USA
e-mail: jonlee@us.ibm.com

$$z(C,s) := \max\{\ln\det C[S,S] \ : \ S \subset N,\ |S| = s\}.$$

This fundamental problem in the *design of experiments* was introduced in [19] and first used in a monitoring design context in [8] (also see [10,17,18,21]).

From the perspective of algorithmic optimization, the problem has been studied extensively; see [1–3,11–13,16] and the surveys [14,15]. Exact algorithms are based on the Branch-and-Bound framework. Crucial to such an approach is a good upper bound on $z(C,s)$. Previous upper bounds are based on either eigenvalues or nonlinear programming relaxations.

In Sect. 2, we describe the masked spectral bound of [3]. In Sect. 3, we describe two first-order rank-restricted algorithms for quickly obtaining a good mask. In Sect. 4, we compare our methods with one another and with the affine-scaling algorithm of [3]. We also examine the issue of how to choose the rank. In Sect. 5, we describe our Branch-and-Bound implementation and results. Finally, in Sect. 6, we mention possibilities for incorporating second-order information, and in Sect. 7, we discuss some future directions.

Notation: For a square symmetric matrix $A$, $A \succeq 0$ (resp., $A \succ 0$) means that $A$ is positive semidefinite (resp., definite). We use $e$ to denote a vector with all components equal to 1. For a square matrix $A$, $\mathrm{diag}(A)$ denotes the vector of diagonal entries from $A$. For a vector $x$, $\mathrm{Diag}(x)$ denotes the square matrix having diagonal $x$ and off-diagonal entries of zero. For a vector $x$, $x^{-1/2}$ denotes the entrywise reciprocal square roots. For $n \times n$ matrices $A$ and $B$, $A \circ B$ denotes Hadamard (i.e., elementwise) product, while $A \bullet B := \mathrm{trace}(AB')$. For a matrix $A$, $A_{i\cdot}$ denotes the $i$th row.

## 2 The masked spectral bound

Anstreicher and Lee introduced the masked spectral (upper) bound (see [3]) for $z(C,s)$. A *mask* is any symmetric, positive semidefinite matrix having ones on the diagonal, i.e., any $X \succeq 0$ having $\mathrm{diag}(X) = e$. We define the associated *masked spectral bound* as

$$\xi_{C,s}(X) := \sum_{l=1}^{s} \ln\left(\lambda_l\left(C \circ X\right)\right),$$

where $\circ$ represents the Hadamard product of matrices and eigenvalues $\lambda_1, \ldots, \lambda_n$ are in nonincreasing order. Validity of the masked spectral bound is based on: (1) *Oppenheim's inequality*, i.e., $\det A \leq \det A \circ B / \prod_{j=1}^{n} B_{jj}$, where $A \succeq 0$ and $B \succ 0$; and (2) the eigenvalue inequalities $\lambda_l(A) \geq \lambda_l(B)$, where $A \succeq 0$, and $B$ is a principal submatrix of $A$.

The masked spectral bound is a generalization of the *spectral partition bound* of [11] (take $X$ to be block diagonal, with blocks of 1's), which itself is a generalization of both the *eigenvalue bound* of [12] (take $X = ee'$) and the *diagonal bound* of [11] (take $X = I$). The spectral partition bound can produce much

better bounds than the eigenvalue and diagonal bounds, but there is no known practical methodology for efficiently choosing a near-optimal partition of $N$, which describes the block structure of the mask (see [11]). Some success has been reported using the following method for calculating the so-called "one-big partition" of $N$ (see [11] and [3]): (1) let $S$ be a heuristic solution of the maximum-entropy sampling problem; (2) the associated *one-big partition* of $N$ has one block $S$ and the elements of $N \setminus S$ as singletons.

The motivation for working with the masked spectral bound is to try to use methods of continuous optimization to get the strength of the combinatorial bounds (i.e., the spectral partition bounds and its specializations) but more efficiently. Specifically, we try to find a mask yielding a good bound. Finding an optimal mask is a problem of minimizing a nondifferentiable nonconvex function subject to a semidefiniteness constraint. That sounds daunting, except for the following inter-related points: (1) for our purposes, we do not need a global minimum; (2) in our experience the nondifferentiability is not so serious; (3) in our experience, local solutions obtained from different starting points are not wildly different in value; and (4) by suitably initializing the search for a mask, we can do at least as well as fixed masks.

An important point is that function, gradient and Hessian evaluations for $\xi$ are expensive. All of these are based on a single (expensive) eigensystem calculation. Function evaluations require eigenvalues, gradients require the eigenvectors as well, and Hessians further require some non-negligible arithmetic. Considering that we are embedding these calculations in Branch-and-Bound, with the goal for each subproblem of trying to push its upper bound below the value of a global lower bound, we cannot afford to spend much time minimizing. So we need to find a way to descend quickly in just a few steps.

Anstreicher and Lee [3] proposed a method for minimizing $\xi$ which was based on ideas coming from the affine scaling algorithm for linear programming. They considered different algorithmic variants, such as short- and long-step approaches, and they demonstrated the ability to achieve good bounds (relative to the one-big, eigenvalue, and diagonal bounds), even in the face of the nonconvexity and nondifferentiability of $\xi$. They did not, however, consider efficient evaluation of the bound or use of the bound in a complete Branch-and-Bound algorithm.

## 3 Rank-restricted masks

Our idea is to work with rank-restricted masks in factored form. That is, we consider masks of the form $X := VV'$, where $V$ is $n \times k$ and $1 \leq k \leq n$. As we try to find a mask yielding a low bound $\xi$, we can hope to make more rapid progress owing to two factors: (1) the semidefiniteness restriction is handled implicitly by the factored form; and (2) by choosing smaller $k$, we work with many fewer variables ($kn$ rather than $n(n-1)/2$), which may lead to faster convergence of a steepest-descent type method and also opens up second-order methods as a

realistic possibility. As it turns out in our experiments, (1) proves to be more beneficial than (2).

We note that the restriction to low-rank PSD matrices has been used successfully in other contexts [4–6]. In fact, the first algorithm that we describe borrows its key ideas from [4].

Our usage of rank-restricted masks will be based on three functions. The first normalizes the rows of $V$, i.e.,

$$h(V) := \mathrm{Diag}\left([\mathrm{diag}(VV')]^{-1/2}\right)V;$$

the second takes a normalized $\bar{V}$ and calculates the corresponding masked spectral bound, i.e.,

$$g(\bar{V}) := \xi_{C,s}(\bar{V}\bar{V}');$$

and the third is the composite function $f(V) := g(h(V))$. Note that $h$ is only well-defined if all rows of $V$ are nonzero.

We propose a steepest-descent type algorithm for minimizing $f(V)$, and for this, we consider the gradient of $f$. Let $u_l(C \circ \bar{V}\bar{V}')$ be an eigenvector, of unit Euclidean norm, associated with $\lambda_l(C \circ \bar{V}\bar{V}')$. Let $U$ be the $n \times s$ matrix having columns $u_l$ $(l = 1, \ldots, s)$, and let $\Sigma$ be the order-$s$ square diagonal matrix with $\Sigma_{ll} = \lambda_l$ $(l = 1, \ldots, s)$. Then, as long as $\lambda_s > \lambda_{s+1}$, we have that the gradient of $f$ at $V$ is the matrix

$$\nabla f(V) = D\,\nabla g(\bar{V}) - (\nabla g(\bar{V})V' \circ D^3)V;$$

where

$$\begin{aligned} d &= \left[\,\mathrm{diag}(VV')\right]^{-1/2}, \\ D &= \mathrm{Diag}(d), \\ \bar{V} &= h(V), \\ \nabla g(\bar{V}) &= 2(C \circ U\Sigma^{-1}U')\bar{V}. \end{aligned}$$

This can be derived using standard results concerning symmetric functions of eigenvalues (see [20], for example). Note that we must define $u_l$ properly when $\lambda_l = \lambda_{l+1}$ for any $l = 1, \ldots, s-1$; in such situations, we just take care that the associated $\{u_l\}$ form an orthonormal basis for each of the eigenspaces corresponding to distinct eigenvalues.

Our steepest-descent algorithm (Algorithm 1) is described in Fig. 1. Due to the possible non-differentiability of $f$, an important ingredient of the algorithm is the acceptance of a non-improving step if descent is not detected after a certain number ($k_{\max}$) of trials. Values such as $k_{\max} := 2$ and $\beta := 1/3$ are practical. (Anstreicher and Lee [3] used a similar back-tracking approach for their affine-scaling algorithm. In particular, they also used parameters $k_{\max} := 2$ and $\beta := 1/3$.) In what follows, we will describe the results of our experiments

0. Initialize $V$, $t_{\max}$, $k_{\max}$ and $0 < \beta < 1$. Set $t = 0$.
1. Let $k := 0$ and $t := t + 1$. If $t > t_{\max}$ then STOP.
2. Let $W := V - \beta^k \nabla f(V)$.
3. If $f(WW') < f(VV')$ or $k = k_{\max}$, then let $V := W$ and GOTO 1. Otherwise, let $k := k + 1$ and GOTO 2.

**Fig. 1** Basic descent algorithm

0. Initialize $\bar{V}$ with unit-length rows, $t_{\max}$, $k_{\max}$ and $0 < \beta < 1$. Set $t = 0$.
1. Let $k := 0$ and $t := t + 1$. If $t > t_{\max}$ then STOP.
2. Let $\bar{W} := h(\bar{V} - \beta^k \nabla g(\bar{V}))$.
3. If $f(\bar{W}\bar{W}') < f(\bar{V}\bar{V}')$ or $k = k_{\max}$, then let $\bar{V} := \bar{W}$ and GOTO 1. Otherwise, let $k := k + 1$ and GOTO 2.

**Fig. 2** Alternative descent algorithm

with various values for $t_{\max}$ and various choices of the initial $V$. For example, in the Branch-and-Bound results of Sect. 5, we will take $t_{\max} \leq 5$.

Several alternatives to Algorithm 1 can be considered. For example, we can define $W := h(V - \beta^k \nabla f(V))$ in Step 2, which normalizes the algorithm's iterates and simplifies the gradient formulas because $d = e$. Another possibility is to move in the direction $-\nabla g(\bar{V})$, i.e., to change Step 2 to read $W := V - \beta^k \nabla g(\bar{V})$. This is valid because, in fact, $-\nabla g(\bar{V})$ is a descent direction for $f$ at $V$:

**Proposition 1** *Suppose that $f$ is differentiable at $V$, and define $\bar{V} := h(V)$. Then $-\nabla g(\bar{V})$ is a descent direction for $f$ at $V$.*

*Proof* Define $M := \nabla g(\bar{V})$, and let $\theta_i$ be the angle between $M_{i\cdot}$ and $V_{i\cdot}$. Then

$$\begin{aligned}
\nabla f(V) \bullet \nabla g(\bar{V}) &= [DM - (MV' \circ D^3)V] \bullet M \\
&= DM \bullet M - (MV' \circ D^3)V \bullet M \\
&= \sum_{i=1}^{n} d_i \|M_{i\cdot}\|^2 - \sum_{i=1}^{n} d_i^3 (V_{i\cdot} M_{i\cdot}')^2 \\
&= \sum_{i=1}^{n} \left( d_i \|M_{i\cdot}\|^2 - d_i \|M_{i\cdot}\|^2 \cos^2 \theta_i \right) \\
&= \sum_{i=1}^{n} d_i \|M_{i\cdot}\|^2 (1 - \cos^2 \theta_i) \\
&\geq 0.
\end{aligned}$$

□

A final possibility is to combine both of these alternatives to define $W := h(V - \beta^k \nabla g(\bar{V}))$. We have experimented with this last alternative, which we refer to as Algorithm 2. It is given in Fig. 2. (The notation $\bar{V}$ and $\bar{W}$ emphasizes that all iterates have unit-norm rows.)

In the next section, we will present evidence to illustrate that the two algorithms above perform quite well, even though they are somewhat simplistic approaches to the minimization of the non-convex, non-differentiable function $\xi$. Overall, we believe that the non-differentiability of $\xi$ is actually not so bad that it precludes the use of simple gradient-based descent methods.

## 4 Experiments with rank-restricted masks

In this section, we detail our experience with Algorithms 1 and 2. A few preliminary remarks are in order. First, the data for our experiments, which has $n = 63$ and was also used in [3], comes from an environmental monitoring application (see [10]).

Second, we will often compare bounds on the *original* and *complementary* problems. This terminology refers to the fact that, using the identity

$$\ln\det C[S,S] = \ln\det C + \ln\det C^{-1}[N \setminus S, N \setminus S], \tag{1}$$

*any* bound for the complementary problem of choosing a maximum entropy set of $n - s$ points with respect to the covariance matrix $C^{-1}$ translates to a bound for the original problem (see [1,2]). In practice, the same class of bound (e.g., the one-big bound) can yield very different values when calculated with respect to the original and complementary problems.

Finally, a primary basis for comparison will be the (absolute) gap between a calculated bound and the entropy value of a heuristic solution. This measure, which is invariant under matrix scaling, has been used in all of the computational experiments reported in previous work on bounds.

The heuristic solution used to calculate the gap is obtained via a routine introduced in [12], which consists of two stages: the greedy construction of a candidate $S$ with $|S| = s$ and a straightforward pairwise-interchange (2-opt) local improvement of $S$. More specifically, the greedy scheme is as follows, where $H(S) := \ln\det C[S,S]$ is the entropy function: initialize $S = \emptyset$, and then, for $j = 1, 2, \ldots, s$, choose $k \in N \setminus S$ so as to maximize $H(S \cup \{k\})$ and adjoin $k$ to $S$. Then beginning from the output set $S$ of the greedy method described above, repeating while possible, choose $k \in N \setminus S$ and $l \in S$ so that $H(S \cup \{k\} \setminus \{l\}) > H(S)$, and replace $S$ with $S \cup \{k\} \setminus \{l\}$.

In fact, this routine can also be applied to the complementary problem via (1) to obtain a second heuristic solution for the original problem. In [12], it has been observed that this second solution is typically of better quality when $s > n/2$, and so we adopt the following scheme: if $s \leq n/2$, then we calculate the heuristic solution via the original problem; otherwise, we calculate it via the complementary problem.

This same scheme for calculating a heuristic solution will also be used in Sect. 5 to generate an initial global lower bound for use in Branch-and-Bound. Also, we did not attempt to improve this heuristic or construct better

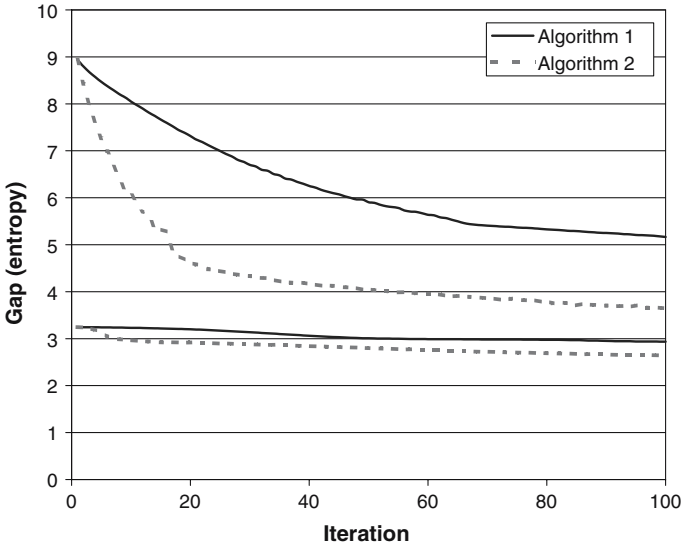


**Fig. 3** Comparison of the gaps produced by Algorithms 1 and 2 when each is run for 100 iterations from the *same, random* starting point on both the *original* and *complementary* problems. Relevant parameters are $(n, s, k) = (63, 31, 63)$. The *top two curves* represent the original problem; the *bottom two curves* represent the complementary problem

alternatives since the main goal of this paper was to evaluate the relative quality of various upper bounds.

### 4.1 Comparison of Algorithms 1 and 2

We first compare Algorithm 1 with Algorithm 2. Because both algorithms require roughly the same amount of work per iteration, we have found that the running times of the algorithms do not differ significantly. Accordingly, we are primarily interested in comparing convergence properties. (For the interested reader, all runs depicted in Fig. 3 below took less than 5 s.)

Although Algorithms 1 and 2 can exhibit different convergence rates or patterns on different instances of the maximum-entropy sampling problem with different values of $k$, we found that, on any specific instance, Algorithm 2 typically converged more quickly and reliably than Algorithm 1. Figure 3 depicts a typical outcome. We ran both algorithms from the same, random starting point (entries uniform in $[-1, 1]$) for 100 iterations with $(n, s, k) = (63, 31, 63)$ on both the original and complementary problems. The curves indicate convergence by depicting entropy gap versus iteration number.

As Fig. 3 demonstrates, Algorithm 2 converges more quickly than Algorithm 1 and also achieves better gaps. This is intriguing behavior in light of the fact that Algorithm 1 employs the steepest descent direction for $f$, while Algorithm 2 uses a kind of projected search path. One hypothesis as to why

we see this behavior with Algorithm 2 is as follows: its search path may be perturbing us away from points where $\xi$ is not differentiable [e.g., Lee and Overton (unpublished) have observed nondifferentiability at true local minimizers obtained by the gradient-sampling algorithm (see [7])].

Another comparison of Algorithms 1 and 2, which supports a similar conclusion as Fig. 3, will be given in Sect. 4.3.1. Nonetheless, we will advocate the use of Algorithm 1 in Sect. 5. Specific reasons for doing so will be explained in that section.

### 4.2 Comparison with affine scaling

We next compare Algorithm 1 with the affine scaling algorithm of Anstreicher and Lee [3], which we refer to as Algorithm AS. In order to obtain as fair a comparison as possible, we obtained their code, which was written in Matlab, and constructed our own Matlab code for Algorithm 1. All tests for both algorithms were initialized with the same starting point as described in [3] (i.e., a positive definite perturbation of a fixed mask). In particular, since Algorithm AS is an interior-point method, we took $k = n$ for Algorithm 1 in all comparisons between the two methods.

Our comparisons are based on the following three criteria, each of which has two possible realizations (for an overall total of eight comparisons):

(a) *initial V* – a full-rank perturbation of either the eigenvalue or one-big mask;
(b) $t_{\max}$ – either 5 or 100;
(c) *problem type* – either original or complementary problem.

In each of the eight comparisons, we ran the algorithms for all $s$ between 3 and 60, inclusive.

The purpose of comparing on the basis of $t_{\max}$ is to observe performance early and late in the algorithm. Early performance (e.g., $t_{\max} = 5$) gives an indication of how the algorithms would perform in a Branch-and-Bound algorithm, where we can only afford to do a few steps to obtain a bound at each node in the tree. On the other hand, late performance (e.g., $t_{\max} = 100$) gives an indication of the overall performance and robustness of the algorithms.

Over all eight comparisons, we found similar behavior, namely that Algorithm 1 did at least as well as – and often significantly better than – Algorithm AS. Accordingly, in the interest of space, we provide two representative comparisons of the eight total in Figs. 4 and 5.

One difference between Algorithms 1 and AS, which we observed but is not immediately evident from Fig. 5, is the ability of Algorithm 1 to achieve lower gaps than Algorithm AS – even when Algorithm AS is allowed to run for a large number of iterations (for example, 1,000 iterations). Said differently, Algorithm AS seems to stall at higher gaps than Algorithm 1.

Another important criterion for comparison is running time. For both Algorithms AS and 1, the work per iteration is dominated by the eigenvalue and eigenvector calculations associated with the function $\xi$. Recall also that both

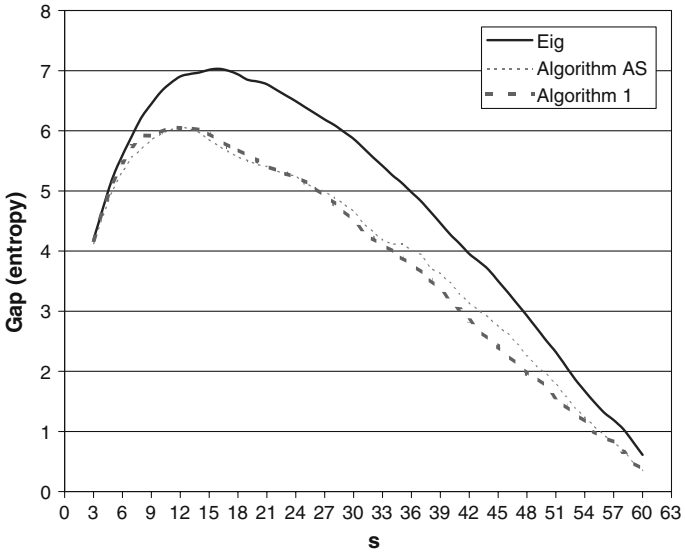


**Fig. 4** Comparison of the gaps produced by Algorithms 1 and AS after five steps when initialized near the *eigenvalue* mask on the *complementary* problem, for $s = 3, \ldots, 60$. The gap given by the eigenvalue mask is also shown for reference

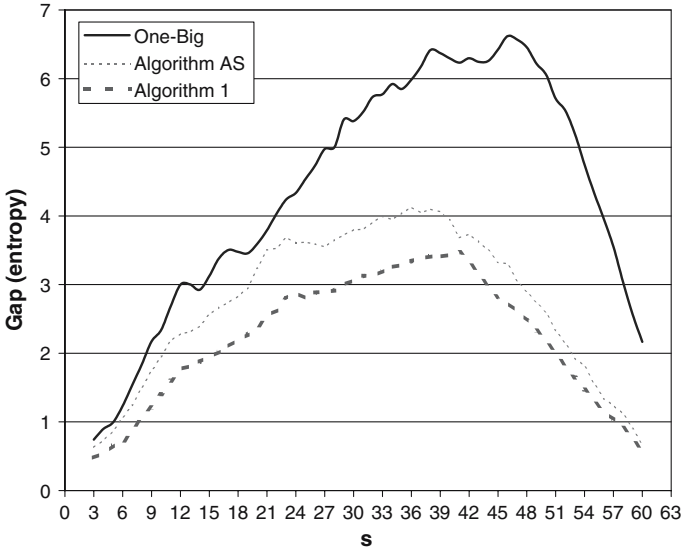


**Fig. 5** Comparison of the gaps produced by Algorithms 1 and AS after 100 steps when initialized near the *one-big* mask on the *original* problem, for $s = 3, \ldots, 60$. The gap given by the spectral partition mask is also shown for reference

algorithms use the same line search back-tracking technique. As such, the running times for the two methods (for a fixed number of iterations) are quite

comparable and so are not a source of differentiation. (For example, each individual run represented in Fig. 5 took less than 5 s.)

Overall, our experiments lead us to favor Algorithm 1 over Algorithm AS. In particular, we will use Algorithm 1 exclusively for our Branch-and-Bound experiments in Sect. 5.

### 4.3 Choosing the rank

An important parametrization issue concerns how the rank $k$ affects the masked spectral bound. In particular, if we can achieve nearly the same bound by running Algorithm 1 with a smaller $k$, we could hope that this would lead to decreased computational requirements. Of course, practically speaking, we need to know precisely which rank to choose and how much computational savings can be expected.

Unfortunately, we have found it difficult to provide any guidelines on which rank to choose. Moreover, the computational savings that we have been able to achieve from reducing the rank are insignificant due to the fact that the eigensystems calculations associated with the function $\xi$ dominate the running of Algorithm 1. (The eigensystems were calculated using Matlab's `eig` command, which in turn calls standard LAPACK subroutines.) We illustrate these points by means of some example runs.

For each $k$ between 2 and 63, we ran Algorithm 1 twice with $s = 31$, once on the original problem and once on the complementary problem. For both runs, we initialized $V$ randomly (entries uniform in $[-1, 1]$) and took $t_{\max} = 1,000$, that is, we did 1,000 iterations. The purpose of so many iterations was to give Algorithm 1 sufficient opportunity to reduce the bound as far as possible. As before, our basis of comparison was the entropy gap. Figure 6 depicts the results.

From Fig. 6, it appears that the optimal rank (that is, the smallest rank that yields the best possible bound) for the original problem is approximately $k = 30$. On the other hand, for the complementary problem, the optimal rank appears to be around $k = 5$. This example illustrates the difficulty in choosing the optimal rank a priori.

For the original problem, the average running time over all $k = 2, \ldots, 63$ was 10.8 s with a standard deviation of 0.3 s; for the complementary problem, the numbers were 24.7 and 1.4, respectively. So, the timings showed little variability with respect to the rank $k$. As mentioned above, this is due to the dominance of the eigensystem calculations.

Overall, a conclusion of our experiments is that, when running Algorithm 1, it is reasonable to take $k = n$ since the resulting running time will not be significantly more than with smaller $k$. (In particular, we did not search further for different strategies of choosing $k$.) On the other hand, our experiments support the choice of lower $k$ to get comparable bounds – if a way can be found to exploit the lower rank for better convergence and/or running times.

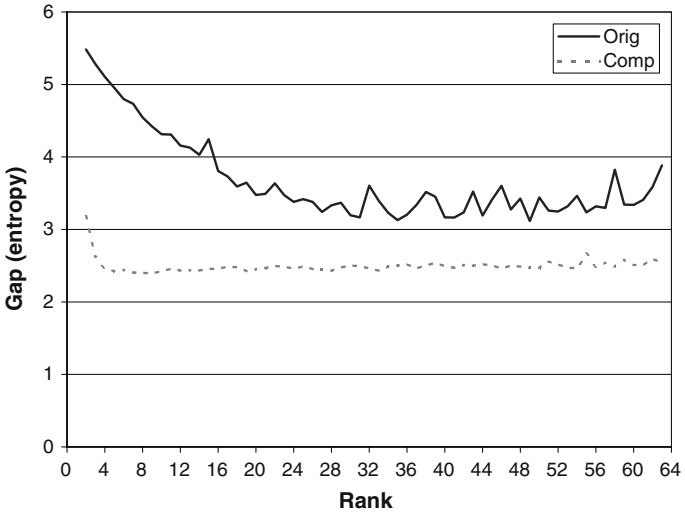


**Fig. 6** Gaps produced by Algorithm 1 after 1,000 steps when initialized *randomly* on the *original* and *complementary* problems, for ranks $k = 2, \ldots, 63$

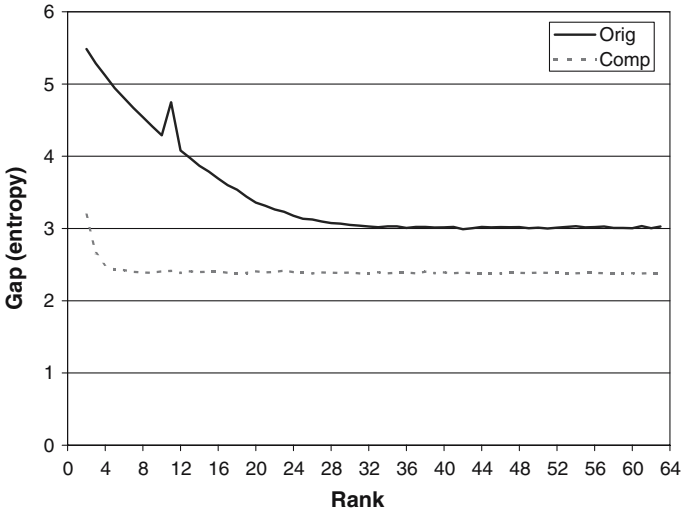


**Fig. 7** Gaps produced by Algorithm 2 after 1,000 steps when initialized *randomly* on the *original* and *complementary* problems, for ranks $k = 2, \ldots, 63$

### *4.3.1 Another comparison of Algorithms 1 and 2*

The above experiments give us an additional opportunity to compare Algorithm 1 with Algorithm 2. In Fig. 7, we replicate the experiments of Fig. 6 except this time with Algorithm 2. Notice that, overall, Algorithm 2 converges more reliably than Algorithm 1 for varying ranks. It is also clear that Algorithm 2

achieves slightly better gaps. This evidence provides additional support for the earlier conclusion that Algorithm 2 is more robust than Algorithm 1.

## 5 Incorporating in Branch-and-Bound

The Branch-and-Bound approach to maximum-entropy sampling was first described in [12]. Branching is done on single elements of $N$ – either forcing a single element into the set $S$ or barring a single element from $S$. Thus, at any node in the Branch-and-Bound tree, the corresponding subproblem is determined by forcing a particular set $F$ of $f$ elements into $S$ and barring a particular set $U$ of $u$ elements from $S$. It remains then to optimize

$$\max\big\{\ln\det C[S,S]\ :\ S\subset N\setminus U,\ F\subset S,\ |S|=s\big\}.$$

By the Schur complement theorem, this problem is in turn equivalent to choosing a set $T$ of $s-f$ elements from the set $N\setminus(U\cup F)$, so as to maximize the conditional entropy $\ln\det C_{F,U}[T,T]$ (plus the constant $\ln\det C[F,F]$), where

$$\begin{aligned}C_{F,U} := {} & C[N\setminus(F\cup U),N\setminus(F\cup U)]\\ & - C[N\setminus(F\cup U),F](C[F,F])^{-1}C[F,N\setminus(F\cup U)].\end{aligned}$$

In other words, the task at each node is to determine $z(C_{F,U},s-f)+\ln\det C[F,F]$, which is itself an instance of the maximum-entropy sampling problem. Hence, any bound developed for the maximum-entropy sampling problem may be used throughout the Branch-and-Bound tree.

We adapted the Branch-and-Bound implementation of [1,2], which was written in the C programming language and uses LAPACK eigensystem calculations. We kept all default options, including the decision rules for selecting the next node in the tree for branching (the node with the largest upper bound) and for selecting the specific index to branch on (the largest index not already in $F$ or $U$). One enhancement that we did make, however, was the calculation of an initial global lower bound via a heuristic solution of the maximum-entropy sampling problem, which has been discussed in Sect. 4.

Our goal was to determine whether optimizing the masked spectral bound at each node of the tree is a useful bounding strategy. We compare with the following bounding strategy (for simplicity, *orig* and *comp* refer to bounds coming from the original and complementary problems, respectively):

*Fixed bounding strategy.* For a fixed type of masked spectral bound (i.e., eigenvalue, diagonal, or one-big), compute *orig* and/or *comp* according to the following steps:

1. If *orig* $<$ *comp* for the parent, then calculate *orig*; otherwise, *comp*.
2. If the calculated bound does not fathom, then also calculate the remaining bound.

Note that, for the eigenvalue bound, *orig* $=$ *comp*, so the above steps simplify.

One consequence of the default bounding strategy is that, high in the tree where fathoming is less likely to occur, it is likely that both *orig* and *comp* will be unnecessarily calculated at each node. We found this drawback to be outweighed by the benefit of incorporating both *orig* and *comp*. In particular, the single-minded strategies of just computing either *orig* or *comp* throughout the tree did not work as well.

In developing our strategy for optimizing the masked spectral bound, we felt that calculating bounds for both the original and complementary problems at each node would be too expensive. On the other hand, we knew it would be highly beneficial to compute the better of the two. After some experimentation, we arrived at a compromise that was based on the following: we often noticed that different subproblems with the same number of elements fixed in and out of $S$ (i.e., the same $f$ and $u$) behaved similarly in terms of which bound, *orig* or *comp*, was stronger. (Though we are unable to provide a full explanation for this behavior, we believe it is not surprising that certain bounds behave similarly when, say, a few elements of $N$ are fixed or when many elements are fixed.) So throughout the Branch-and-Bound calculations, we kept track of all pairs $(f, u)$. The first time that a subproblem with a particular $(f, u)$ was encountered, we calculated both *orig* and *comp* and took note of which was stronger. Afterwards, every time the same $(f, u)$ was observed, we would calculate *orig* or *comp* according to our first experience.

Our overall strategy for optimizing the masked spectral bound at each node is as follows:

*Optimization bounding strategy.* Determine whether to compute *orig* and/or *comp* (as described above). Then, for each bound to be calculated, do the following:

1. Run Algorithm 1 with $t_{\max} = 2$, terminating immediately if fathoming occurs.
2. Using the points generated by Algorithm 1 so far (three points, including the starting point), use quadratic interpolation to judge whether fathoming will occur in the next three iterations.
3. If so, then set $t_{\max} = 5$ and continue Algorithm 1, terminating immediately if fathoming occurs; otherwise, stop.

We note that the use of quadratic interpolation is a simple, reasonably effective way to judge how quickly the objective function is descending.

A couple of other details concerning the optimization bounding strategy are worth mentioning. First, in accordance with our discussion in Sect. 4, we take full rank, i.e., $k = s - f$, in Algorithm 1 at each node of the tree. Second, instead of initializing $V$ at each node randomly, we attempted to simulate the warm starting of a child node by its parent. We did this by keeping a single, shared-memory copy of $V$, which was initialized randomly and made available to all nodes in the tree. A subproblem of size $s - f$ corresponding to the matrix $C_{F,U}$ (defined above) was then initialized from the submatrix $V[N \setminus (U \cup F), N \setminus (U \cup F)]$, and its final iterate was stored back into the same entries of $V$. Although heuristic, we found this warm-start approach to be much better than our original strategy

of initializing each node randomly. (The root node was initialized with $V$ having entries uniform in $[-1, 1]$.)

Even though the above strategy uses Algorithm 1, it could also be based on Algorithm 2. In Branch-and-Bound, however, we advocate the use of Algorithm 1, even though Sect. 4 suggests the superiority of Algorithm 2. Our decision is based on experiments, which showed that the performance of Algorithm 2 in Branch-and-Bound was worse in almost all cases. (One exception having $(n, s) = (63, 31)$ is given in the next paragraph.) Although the underlying reasons for this behavior are not entirely clear to us, we did notice certain characteristics of the Branch-and-Bound runs, which indicated that Algorithm 2 was doing more work on average than Algorithm 1 (apparently without significant improvement in bounds). For example, Algorithm 2 required about 2.5 trial stepsizes per linesearch on average, while Algorithm 1 required about 1.05.

During early experiments with Branch-and-Bound, it became clear to us that none of the bounding strategies that we have proposed would be able to solve the hardest instances (e.g., $s = 31$ for the data with $n = 63$) in a reasonably short period of time (say, a few hours). In this sense, we cannot claim that optimizing the masked spectral bound is the "silver bullet" for solving extremely large and difficult instances. (It is worth mentioning that we were able to find and prove optimality for the $(n, s) = (63, 31)$ problem in 84 h using Algorithm 1 – and 65 h using Algorithm 2 – on a Pentium 4 2.4 GHz running Linux. This problem is out of reach using a fixed mask, and in fact, this is the first time that this instance has been solved using a strategy based completely on eigenvalue-based bounds.)

Nevertheless, it was also clear that the various bounding strategies behaved very differently. In order to highlight these differences while maintaining a reasonable testing environment, we settled upon the following testing scheme. For a particular instance of the maximum-entropy sampling problem, we ran each of the four bounding strategies – the three fixed bounding strategies and the optimized bounding strategy – for at most 3,600 s. We then calculate the gap between the global upper bound (i.e., the largest upper bound of all nodes remaining in the tree) and the global lower bound (i.e., the entropy of the best feasible solution found so far).

Since we limit the Branch-and-Bound calculations to 3,600 s, we stress that the computational results shown in Figs. 8 and 9 below should be interpreted simply as an indication that our approach is more effective in reducing the gaps when compared with other bounding strategies based on masks. This is not to say that it is the most effective among all bounding strategies or that all problems considered here will be solved to optimality with modest increases in time. For example, a completely different class of bounds based on nonlinear programming techniques have been shown in [2] to be quite effective on the $n = 63$ instances, for example solving the $s = 31$ instance in 3,300 s (when the computer clock speed of 125 MHz in [2] is normalized to our clock speed of 2.4 GHz). For the $n = 124$ instances shown in Fig. 9, it should also be mentioned

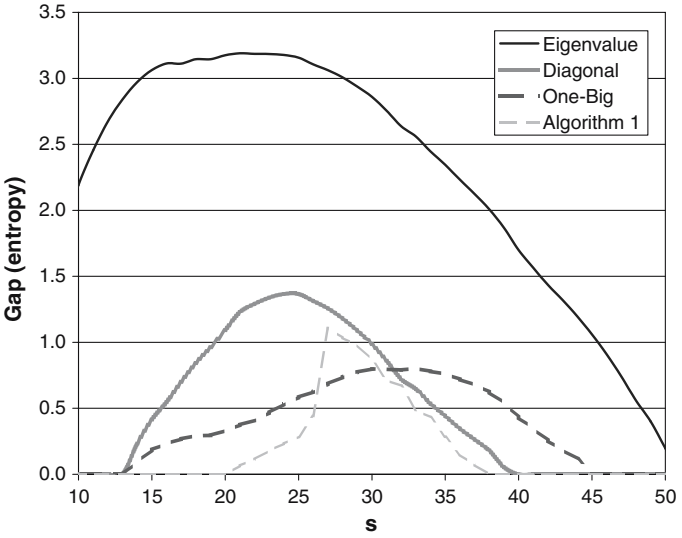


**Fig. 8** Branch-and-Bound results with $n = 63$ after 3,600 s for the four bounding strategies. Gap refers to the difference between global upper and lower bounds at termination

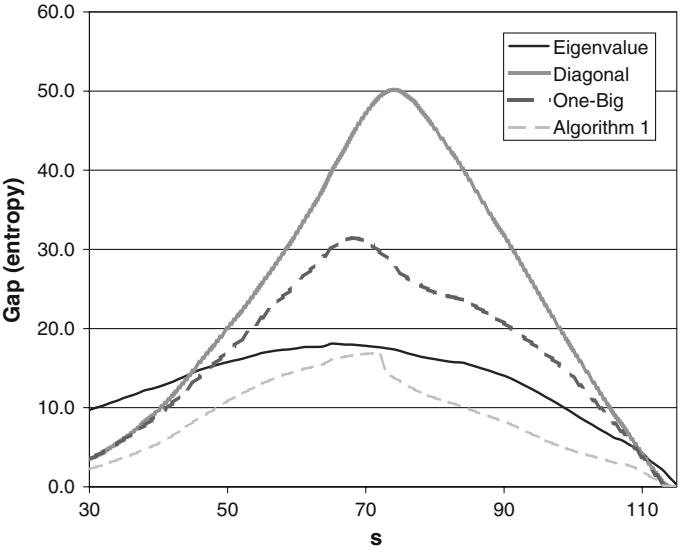


**Fig. 9** Branch-and-Bound results with $n = 124$ after 3,600 s for the four bounding strategies. Gap refers to the difference between global upper and lower bounds at termination

that no known bounding technique has been able to solve the hardest of these instances.

Figure 8 gives Branch-and-Bound results for the data with $n = 63$ introduced in Sect. 4. The results for each of the four bounding strategies are graphed for $s = 10, \ldots, 50$. From this figure, we can see that the optimization bound

strategy did uniformly better than the fixed bound strategies based on the eigenvalue and diagonal masks. Furthermore, compared to the one-big spectral partition bound, our method performed better in the ranges $s = 14, \ldots, 19$ and $s = 30, \ldots, 44$.

In Fig. 9, we give similar results for a data set with $n = 124$, which was first used in [16]. Again, we see that optimizing the masked spectral bound is quite competitive with the other bounding strategies.

## 6 Exploiting second-order information

As mentioned at the beginning of Sect. 3, one of our initial motivations for considering rank-restricted masks was the opportunity for reducing the dimension of the problem so that second-derivative knowledge of $\xi$ could be incorporated efficiently.

In particular, we were interested in developing a variant of Algorithm 1 with search directions produced by a modified Newton's method with exact Hessians. We did successfully implement just such a method and tested it with small rank, e.g., $k \approx 5$, but unfortunately, it did not outperform the steepest-descent version of Algorithm 1. Downsides to the method included the time required to form the Hessian and to factor the modified Hessian as well as the reduced bound quality caused by taking $k$ small. In addition, the strong nonconvexity displayed by $\xi$ resulted in search directions that were not much better than steepest descent.

We also tried other approaches for incorporating second-order information, including BFGS and limited-memory BFGS search directions. Here again, however, our attempts at improving steepest descent were unsuccessful. An added complication was a more stringent strong-Wolfe linesearch (see [9], for example) required by BFGS-type directions. (Since the first version of this paper appeared, recent results obtained by Overton (private communication) suggest that the strong-Wolfe linesearch should not be used when applied within BFGS for the optimization of nondifferentiable functions.)

We include our experiences here in order to provide a complete picture of our efforts and a starting point for additional research. We also include below the exact Hessian formulae for $\xi$ since these have not appeared before in the literature and since we are hopeful that they will be of use to others. The formulae were developed from [20].

In the results below, all eigenvectors are taken to have unit length. Note that, as in the development of the gradient of $f(V)$, we must define $u_l$ properly when $\lambda_l = \lambda_{l+1}$.

We first examine the Hessian of $\xi_{C,s}(X)$. For simplicity, and with an eye toward implementation, we consider the symmetric variable $X$ to be encoded as a column vector of dimension $n(n+1)/2$, where the lower-triangular part of $X$ is stored in column-major format. In the theorem below, the vectors $\sigma_{kl}$ are indexed by the entries of $X$ in the same fashion.

**Proposition 2** *Suppose $X \succeq 0$, and let $\{(\lambda_k, u_k)\}_{k=1}^n$ be the eigenvalues and eigenvectors of $C \circ X$. Suppose also that $\lambda_s(C \circ X) > \lambda_{s+1}(C \circ X)$. Then $\xi$ is analytic at $X$ with Hessian*

$$\nabla^2 \xi_{C,s}(X) = \sum_{k=1}^{s} \sum_{l=1}^{s} \left( -\frac{1}{\lambda_k \lambda_l} \right) \sigma_{kl} \sigma_{kl}' + 2 \sum_{k=1}^{s} \sum_{l=s+1}^{n} \left( \frac{1}{\lambda_k} \frac{1}{\lambda_k - \lambda_l} \right) \sigma_{kl} \sigma_{kl}',$$

*where, for each $(k, l)$,*

$$\left[\sigma_{kl}\right]_{ij} = \begin{cases} \left[C \circ u_l u_k'\right]_{ij} + \left[C \circ u_l u_k'\right]_{ji} & \text{if } i > j \\ \left[C \circ u_l u_k'\right]_{ij} & \text{if } i = j. \end{cases}$$

We next examine the Hessian of $f(V)$. It is helpful here as well to consider $V$ to be encoded as a column vector in column-major format.

**Proposition 3** *Suppose each row of $V$ is nonzero, and define*

$$\begin{aligned} d &:= \left[\operatorname{diag}(VV')\right]^{-1/2}, \\ \bar{V} &:= h(V), \\ X &:= \bar{V}\bar{V}'. \end{aligned}$$

*Let $\{(\lambda_k, u_k)\}_{k=1}^n$ be the eigenvalues and eigenvectors of $C \circ X$, and define*

$$H := \sum_{k=1}^{s} \lambda_k^{-1} u_k u_k'.$$

*Suppose also that $\lambda_s(C \circ X) > \lambda_{s+1}(C \circ X)$. Then $f$ is analytic at $V$ with Hessian*

$$\begin{aligned} \nabla^2 f(V) = &\sum_{k=1}^{s} \sum_{l=1}^{s} \left( -\frac{1}{\lambda_k \lambda_l} \right) \sigma_{kl} \sigma_{kl}' + 2 \sum_{k=1}^{s} \sum_{l=s+1}^{n} \left( \frac{1}{\lambda_k} \frac{1}{\lambda_k - \lambda_l} \right) \sigma_{kl} \sigma_{kl}' \\ &+ \mathcal{M} + \mathcal{N} + \mathcal{P}, \end{aligned}$$

*where, for all $(k, l)$ [and defining $J := C \circ \left(u_l u_k' + u_k u_l'\right)$ and $\bar{x} := \bar{V}_{\cdot p}$ locally],*

$$\left[\sigma_{kl}\right]_{ip} = \left[d \circ (J\bar{x} - \bar{x} \circ (J \circ X)\, e)\right]_i$$

*and where, for all $(ip, jq)$ [and defining $\bar{x} := \bar{V}_{\cdot p}$ and $\bar{y} := \bar{V}_{\cdot q}$ locally],*

$$\begin{aligned} \mathcal{M}_{(ip)(jq)} &= 2\left[dd' \circ \left((e_p' e_q) ee' - (\bar{x} \circ \bar{y})e' - e(\bar{x} \circ \bar{y})' + \bar{x}\bar{y}' \circ X\right) \circ H\right]_{ij}, \\ \mathcal{N}_{(ip)(jq)} &= \begin{cases} 2\left[d^2 \circ (3\bar{x} \circ \bar{y} \circ (X \circ H)e - \bar{x} \circ H\bar{y} - \bar{y} \circ H\bar{x})\right]_i & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \end{aligned}$$

*and*

$$\mathcal{P}_{(ip)(jq)} = \begin{cases} -2\left[d^2 \circ (X \circ H)e\right]_i & \text{if } (i,j) = (p,q) \\ 0 & \text{otherwise.} \end{cases}$$

## 7 Further directions

One possible extension would be to combine our low-rank approach with gradient-sampling ideas (see [7]) when nondifferentiability becomes an issue at a mask that nearly fathoms, though it may well be that the time would be just as well spent on further children.

We feel that exploiting second-order information, particularly when working with low-rank masks, is still an interesting avenue of research. On some smaller instances (e.g., $n \approx 50$) of Branch-and-Bound with Algorithm 1, we noticed that taking low rank produced overall faster running times than taking full rank. At this time, we are unable to reconcile this behavior with our observations concerning rank in Sect. 4, but it would be interesting if one could determine effective strategies for using small rank – and then effectively incorporate second-order information as well.

**Acknowledgements** The authors are grateful to Kurt Anstreicher for introducing them to one another and for providing many insightful comments on the paper. Michael Overton provided invaluable input as well.

## References

1. Anstreicher, K.M., Fampa, M., Lee, J., Williams, J.: Continuous relaxations for constrained maximum-entropy sampling. In: Integer Programming and Combinatorial Optimization (Vancouver, BC, 1996). Lecture Notes in Computer Science, vol. 1084, pp. 234–248. Springer, Berlin Heidelberg New York (1996)
2. Anstreicher, K.M., Fampa, M., Lee, J., Williams, J.: Using continuous nonlinear relaxations to solve constrained maximum-entropy sampling problems. Math. Program. Ser. A **85**(2), 221–240 (1999)
3. Anstreicher, K.M., Lee, J.: A masked spectral bound for maximum-entropy sampling. In: mODa 7–Advances in Model-Oriented Design and Analysis, Contributions to Statistics, pp. 1–10. Physica-Verlag, Heidelberg (2004)
4. Burer, S., Monteiro, R.D.C.: A projected gradient algorithm for solving the Maxcut SDP relaxation. Optim. Methods Softw. **15**, 175–200 (2001)
5. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program. Ser. B **95**(2), 329–357 (2003)
6. Burer, S., Monteiro, R.D.C., Zhang, Y.: Rank-two relaxation heuristics for max-cut and other binary quadratic programs. SIAM J. Optim. **12**(2), 503–521 (2001)
7. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. (to appear) (2003)
8. Caselton, W.F., Zidek, J.V.: Optimal monitoring networks. Stat. Probab. Lett. **2**, 129–178 (1984)
9. Fletcher, R.: Practical Methods of Optimization, 2nd edn. Wiley, New York (1987)
10. Guttorp, P., Le, N.D., Sampson, P.D., Zidek, J.V.: Using entropy in the redesign of an environmental monitoring network. Technical report 116, Department of Statistics, University of British Columbia (1992)
11. Hoffman, A., Lee, J., Williams, J.: New upper bounds for maximum-entropy sampling. In: Atkinson, A.C., Hackl, P., Müller, W.G. (eds.) MODA 6 – Advances in Model-Oriented Design and Analysis, pp. 143–153. Springer, Berlin Heidelberg New York (2001)

12. Ko, C.W., Lee, J., Queyranne, M.: An exact algorithm for maximum entropy sampling. Oper. Res. **43**(4), 684–691 (1995)
13. Lee, J.: Constrained maximum-entropy sampling. Oper. Res. **46**(5), 655–664 (1998)
14. Lee, J.: Semidefinite programming in experimental design. In: Henry Wolkowicz, R.S., Vandenberghe, L. (eds.) Handbook of Semidefinite Programming, International Series in Operations Research and Managemant Science, vol. 27, pp. 528–532. Kluwer, Boston (2000)
15. Lee, J.: Maximum-entropy sampling. In: El-Shaarawi, A.H., Piegorsch, W.W. (eds.) Encyclopedia of Environmetrics, vol. 3, pp. 1229–1234. Wiley, New York (2001)
16. Lee, J., Williams, J.: A linear integer programming bound for maximum-entropy sampling. Math. Program. Ser. B **94**(2–3), 247–256 (2003)
17. Müller, W.G.: Collecting Spatial Data: Contributions to Statistics (Optimum Design of Experiments for Random Fields), revised edn. Physica-Verlag, Heidelberg (2001)
18. Sebastiani, P., Wynn, H.P.: Maximum entropy sampling and optimal Bayesian experimental design. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **62**(1), 145–157 (2000)
19. Shewry, M.C., Wynn, H.P.: Maximum entropy sampling. J. Appl. Stat. **46**, 165–170 (1987)
20. Tsing, N.K., Fan, M.K.H., Verriest, E.I.: On analyticity of functions involving eigenvalues. Linear Algebra Appl. **207**, 159–180 (1994)
21. Wu, S., Zidek, J.V.: An entropy based review of selected NADP/NTN network sites for 1983–1986. Atmos. Environ. **26A**, 2089–2103 (1992)