Math. Program., Ser. B 94: 247–256 (2003)

Digital Object Identifier (DOI) 10.1007/s10107-002-0318-x

Jon Lee · Joy Williams

A linear integer programming bound for maximum-entropy sampling

Received: September 27, 2000 / Accepted: July 26, 2001
Published online: September 5, 2002 –

Abstract. We introduce a new upper bound for the maximum-entropy sampling problem. Our bound is described as the solution of a linear integer program. The bound depends on a partition of the underlying set of random variables. For the case in which each block of the partition has bounded cardinality, we describe an efficient dynamic-programming algorithm to calculate the bound. For the very special case in which the blocks have no more than two elements, we describe an efficient algorithm for calculating the bound based on maximum-weight matching. This latter formulation has particular value for local-search procedures that seek to find a good partition. We relate our bound to recent bounds of Hoffman, Lee and Williams. Finally, we report on the results of some computational experiments.

Key words. experimental design – design of experiments – entropy – maximum-entropy sampling – matching – integer program – spectral bound – Fischer's inequality – branch-and-bound – dynamic programming

Introduction

The maximum-entropy sampling problem is a fundamental problem in the *design of experiments*. The problem has practical application in the design of monitoring networks (see Ko, Lee and Queyranne [8] and the references therein).

Let C be an $n \times n$ real symmetric positive definite matrix with row (and column) indices $N = \{1, 2, \ldots, n\}$. For $S, T \subset N$, we let $C[S, T]$ denote the submatrix of C with row indices S and column indices T. We let $\det C[S, S]$ denote the determinant of the principal submatrix $C[S, S]$, with the convention that $\det C[\emptyset, \emptyset] = 1$. Let s be an integer satisfying $0 \leq s \leq n$. The *maximum-entropy sampling problem* is to solve

$$z := \max_{\substack{S \subset N: \\ |S|=s}} \ln \det C[S, S] \, .$$

In the context of experimental design, C is a covariance matrix of a set of random variables having a joint Gaussian distribution, and, up to some constants, $\ln \det C[S, S]$

Jon Lee: Department of Mathematical Sciences, Thomas J. Watson Research Center, IBM, P.O. Box 218, Yorktown Heights, New York 10598 USA, e-mail: jonlee@us.ibm.com

Joy Williams: Avaya, Inc. 1300 West 120th Avenue, Westminster, Colorado 80234 USA, e-mail: jdwilliams@avaya.com

Mathematics Subject Classification (2000): 52B12, 90C10

Send offprint requests to: Jon Lee

Correspondence to: Jon Lee

is the entropy of the set of random variables indexed by the elements of S. So the maximum-entropy sampling problem amounts to finding a most-informative s-subset of a set of n random variables having a (joint) Gaussian distribution. In the application to the design of environmental monitoring networks, the random variables correspond to levels of acid rain at monitoring locations.

The maximum-entropy sampling problem is NP-Hard, and exact algorithms are based on the branch-and-bound framework with upper bounds calculated by a variety of methods (see Ko, Lee and Queyranne [8], Anstreicher, Fampa, Lee and Williams [1, 2], Lee [9–11], and Hoffman, Lee and Williams [6]). Branching consists of fixing an index j out of the solution (deleting row and column j of C) or fixing an index j in to the solution (pivoting in C on C_{jj}, and decrementing s by one. Lower bounds for limiting the branching are determined by heuristic search methods. See [8, 9, 1, 2] for more details.

In [1, 2, 6], we described how different upper bounds can be calculated by considering a "complementary problem". We have assumed that C is positive definite, so it is nonsingular. Since

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

we have the *complementary problem*

$$z = \ln\det C + \max_{\substack{N\setminus S\subset N:\\ |N\setminus S|=n-s}} \ln\det C^{-1}[N\setminus S, N\setminus S]\,.$$

We can calculate an upper bound with respect to choosing the $n-s$ element set $N\setminus S$ for this problem, and then just add $\ln\det C$ to that upper bound.

In Section 1, we develop a new upper bound on z which is parameterized by a partition of N. The new bound is described as the optimal value of a linear integer program. For the case in which each block of the partition has bounded cardinality, we describe an efficient dynamic-programming algorithm to calculate the bound. In Section 2, we demonstrate that when each block of the partition has no more than two elements, we can calculate the bound by solving a maximum-weight matching problem. This is particularly relevant for local-search procedures that seek to find a good partition (also see Section 4). In Section 3, we compare the new bound to other bounds. In Section 4, we make some remarks regarding the problem of finding a good partition, and we report on results of some computational experiments.

1. A linear integer programming bound

Let p ($\leq n$) be a positive integer, and let $\mathcal{N} = (N_1, N_2, \ldots, N_p)$ be a partition of N into p nonempty parts. For $i = 1, 2, \ldots, p$ and $k = 0, 1, \ldots, |N_i|$, let

$$f_k(N_i) := \max\{\ln\det C[S,S] \;:\; S\subset N_i,\ |S| = k\}\,.$$

We emphasize that $f_0(N_i) = 0$, and we note that $f_s(N) = z$. Also, the assumption of C being positive definite implies that $\det C[S,S] > 0$ for all $S\subset N$; so the $f_k(N_i)$ are well defined.

From a practical point of view, we assume that N is large and s is neither small nor large; so it is impractical to calculate z by enumerating the s-subsets of N. On the other

hand, we assume that each N_i is small enough to enable us to calculate $f_k(N_i)$, for all values, by enumeration. Indeed, in the next section we will focus on the special case for which we have $|N_i| \leq 2$, for $i = 1, 2, \ldots, p$.

Next, with the $f_k(N_i)$ defined, we introduce a new *linear integer programming bound*

$$g_s(\mathcal{N}) := \max \sum_{i=1}^{p} \sum_{k=1}^{|N_i|} f_k(N_i) x_k(i)$$

$$\text{subject to } \sum_{k=1}^{|N_i|} x_k(i) \leq 1, \text{ for } i = 1, 2, \ldots, p; \tag{1}$$

$$\sum_{i=1}^{p} \sum_{k=1}^{|N_i|} k x_k(i) = s; \tag{2}$$

$$x_k(i) \in \{0, 1\}, \text{ for } i = 1, 2, \ldots, p, \quad k = 1, 2, \ldots, |N_i|. \tag{3}$$

Before we establish that $g_s(\mathcal{N})$ is in fact an upper bound on z, it is convenient to review a related bound. Hoffman, Lee and Williams [6] introduced the spectral partition bound which is also based on a partition $\mathcal{N}$ of N. Let $\lambda_l(N_i)$ denote the l^{th} greatest element of the multiset of $|N_i|$ eigenvalues of $C[N_i, N_i]$. Let $\Lambda_l(\mathcal{N})$ denote the l^{th} greatest element from the multiset union of the n eigenvalues of the p matrices $C[N_i, N_i]$. The *spectral partition bound* is

$$\phi_s(\mathcal{N}) := \sum_{l=1}^{s} \ln \Lambda_l(\mathcal{N}).$$

Proposition 1. $z \leq g_s(\mathcal{N}) \leq \phi_s(\mathcal{N})$.

Proof. Let S be any s-subset of N. For $i = 1, 2, \ldots, p$ and $k = 1, 2, \ldots, |N_i|$, let

$$x_k(i) := \begin{cases} 1, & \text{if } |N_i \cap S| = k; \\ 0, & \text{if } |N_i \cap S| \neq k. \end{cases}$$

The objective value of this solution in the linear integer program is

$$\sum_{i=1}^{p} f_{|N_i \cap S|}(N_i),$$

which is at least

$$\sum_{i=1}^{p} \ln \det C[N_i \cap S, N_i \cap S], \tag{4}$$

which is at least

$$\ln \det C[S, S],$$

by Fischer's inequality (see [4, pp. 36–7] or [7, pp. 478–9]). Also, (4) is no more than $\phi_s(\mathcal{N})$, since the eigenvalue interlacing property (see [7, pp. 185–6]) implies that

$$f_k(N_i) \leq \sum_{l=1}^{k} \ln \lambda_l(N_i).$$

□

An extreme case of the spectral partition bound is when $\mathcal{N}$ is a partition of N into singletons. In this case we obtain Hoffman, Lee and Williams' *diagonal bound*

$$\begin{aligned}\psi_s^1 &:= \phi_s(\{1\}, \{2\}, \ldots, \{n\}) \\ &= g_s(\{1\}, \{2\}, \ldots, \{n\}) \\ &= \sum_{l=1}^{s} \ln C_{[ll]},\end{aligned}$$

where $C_{[ll]}$ denotes the l^{th} greatest diagonal entry of C. Since the linear integer programming bound can not decrease as we refine the partition, we easily have the following result.

Proposition 2. *For every partition $\mathcal{N}$ of N, $g_s(\mathcal{N}) \leq \psi_s^1$.*

Even though the linear integer programming bound dominates the spectral partition bound, the linear integer programming bound may not be practical to calculate if some of the N_i are too large to enable us to compute some $f_k(N_i)$. If we want to allow the possibility of it being impractical to calculate $f_k(N_i)$ for *some* values of k (for example, when N_i is large and k is middling), we can, for *those* values, replace $f_k(N_i)$ with *any* upper bound on $f_k(N_i)$ (for example, the sum of the logarithms of the k greatest eigenvalues of $C[N_i, N_i]$ as the spectral partition bound does, or any other upper bound – see [10, 11], for example) – in fact, we take this approach in our computational experiments (see Section 4).

Next, we assume that we can calculate the $f_k(N_i)$ (for example, if the N_i have cardinality bounded by a (small) constant), and we describe an efficient dynamic-programming algorithm to calculate $g_s(\mathcal{N})$. We define the value function

$$\begin{aligned}v_t(j) := \max & \sum_{i=1}^{j} \sum_{k=1}^{|N_i|} f_k(N_i) x_k(i) \\ & \text{subject to } \sum_{k=1}^{|N_i|} x_k(i) \leq 1, \text{ for } i = 1, 2, \ldots, j; \\ & \sum_{i=1}^{j} \sum_{k=1}^{|N_i|} k x_k(i) = t; \\ & x_k(i) \in \{0, 1\}, \text{ for } i = 1, 2, \ldots, j, \\ & \qquad\qquad k = 1, 2, \ldots, |N_i|,\end{aligned}$$

for $j = 1, 2, \ldots, p$, and $t = 0, 1, \ldots, \min\{\sum_{i=1}^{j} |N_i|, s\}$, and we let $v_t(j) := -\infty$ when $\sum_{i=1}^{j} |N_i| < t \leq s$ (i.e., when the linear integer program is infeasible). The form of the value function is motivated by the observation that $g_s(\mathcal{N}) = v_s(p)$. Using the boundary conditions

$$v_t(0) := \begin{cases} 0, & \text{for } t = 0; \\ -\infty, & \text{for } t = 1, 2, \ldots, s, \end{cases}$$

we can calculate $g_s(\mathcal{N}) = v_s(p)$ efficiently, by using the recursion

$$v_t(j) = \max_{0 \le k \le \min\{|N_j|, t\}} \left\{ f_k(N_j) + v_{t-k}(j-1) \right\}.$$

Finally, there is the possibility of just solving the linear-programming relaxation of the linear integer program, to obtain a weaker, but possibly more easily computed bound. This may be useful if N is extremely large (but the N_i are small). In addition, there is the potential to efficiently resolve the linear-programming problems after small changes to the data.

2. Blocks with no more than two elements

Now, we assume that $|N_i| \le 2$, for $i = 1, 2, \ldots, p$. In this special case, we provide an efficient algorithm, based on matchings, for calculating the linear integer programming bound $g_s(\mathcal{N})$. This may be particularly useful in the context of local-search procedures for finding a good partition (see Section 4), since there is the potential to efficiently resolve the matching problems after small changes to the data.

Our "matchings" below consist of assignments of integers to the edges of a graph so that certain integer upper/lower vertex capacities and integer upper/lower edge bounds are respected. Note that we do not assume that any of these integers are nonnegative.

We define a simple undirected graph $G_s(\mathcal{N})$. The vertices consist of $i = 1, 2, \ldots, p$, and two special vertices Z ("zero") and T ("two"). Each vertex $i = 1, 2, \ldots, p$ has lower capacity $\underline{b}(i) := 0$ and upper capacity $\overline{b}(i) := 1$. Vertex Z has lower capacity $\underline{b}(Z) := 0$ and upper capacity $\overline{b}(Z) := 0$. Vertex T has lower capacity $\underline{b}(T) := s - p$ and upper capacity $\overline{b}(T) := s - p$.

For each $i = 1, 2, \ldots, p$, there is an edge between i and Z, with lower capacity $\underline{c}(iZ) := 0$, upper capacity $\overline{c}(iZ) := 1$, and weight $w(iZ) := -f_1(N_i)$.

For each $i = 1, 2, \ldots, p$ such that $|N_i| = 2$, there is an edge between i and T, with lower capacity $\underline{c}(iT) := 0$, upper capacity $\overline{c}(iT) := 1$, and weight $w(iT) := f_2(N_i) - f_1(N_i)$.

Finally, there is an edge between Z and T, with lower capacity $\underline{c}(ZT) := -\infty$, upper capacity $\overline{c}(ZT) := +\infty$, and weight $w(ZT) := 0$.

In Figure 1, we depict the graph $G_s(\mathcal{N})$ together with its vertex and edge lower and upper capacities as ordered pairs. The idea is that selecting the edge (iZ) (respectively, (iT)) for the matching corresponds to picking zero (respectively, two) elements from N_i. If neither (iZ) nor (iT) is selected for the matching, then the "default" is that one element is picked from N_i. The vertex capacities at Z and T are set so that, with the help of the edge (ZT), exactly s elements are picked in total. All of this is formalized in the proof of Proposition 3 below.

Let $\mu_s(\mathcal{N})$ be the constant $\sum_{i=1}^{p} f_1(N_i)$ plus the maximum weight of an assignment of numbers $y(e)$ to the edges e of $G_s(\mathcal{N})$ satisfying the *c-capacitated b-matching constraints*

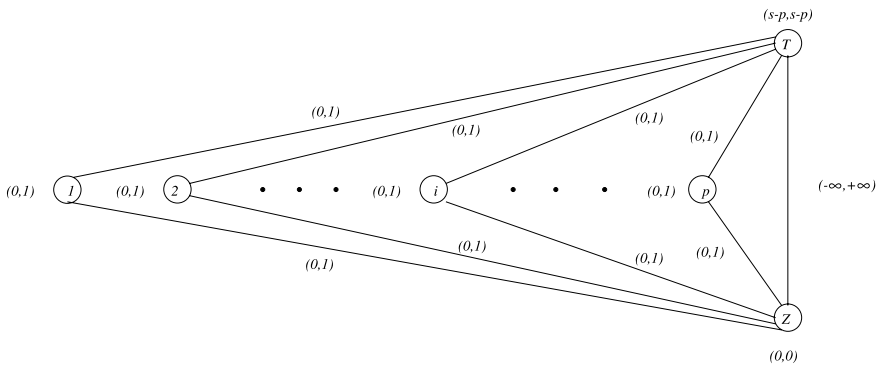

Fig. 1. The capacitated graph $G_s(\mathcal{N})$

$$\underline{c}(e) \le y(e) \le \overline{c}(e) \text{ and integer, for all edges } e;$$
$$\underline{b}(v) \le \sum\{y(e) \ : \ e \text{ meets } v\} \le \overline{b}(v), \text{ for all vertices } v.$$

We note that a maximum-weight c-capacitated b-matching can be found in polynomial time (see Edmonds and Johnson [3]; also see Schrijver [12, Section 6, "Matchings and generalizations"] for a nice discussion of the subject). Therefore, we can efficiently calculate $\mu_s(\mathcal{N})$.

Proposition 3. *Let $\mathcal{N}$ be a partition of N with $|N_i| \le 2$, for $i = 1, 2, \ldots, p$. Then $\mu_s(\mathcal{N}) = g_s(\mathcal{N})$.*

Proof. Suppose that the numbers $y(e)$ solve the c-capacitated b-matching program that defines $\mu_s(\mathcal{N})$. For $i = 1, 2, \ldots, p$, let

$$x_2(i) := y(iT), \text{ for } i \text{ such that } |N_i| = 2;$$
$$x_1(i) := \begin{cases} 1 - y(iZ) - y(iT), & \text{for } i \text{ such that } |N_i| = 2; \\ 1 - y(iZ), & \text{for } i \text{ such that } |N_i| = 1. \end{cases}$$

The edge capacities on iZ and iT and the vertex capacities on i ensure that $y(iZ)$ and $y(iT)$ are binary with at most one of them equal to 1. Then, the choice of $x_1(i)$ and $x_2(i)$ dictates that they too are binary and that at most one of them is equal to 1. So, (1) and (3) are satisfied. Now,

$$\begin{aligned} \sum_{i=1}^{p} \sum_{k=1}^{|N_i|} k x_k(i) &= \sum_{i=1}^{p} x_1(i) + 2 \sum_{i:|N_i|=2} x_2(i) \\ &= \sum_{i:|N_i|=2} (1 - y(iZ) - y(iT)) + \sum_{i:|N_i|=1} (1 - y(iZ)) + 2 \sum_{i:|N_i|=2} y(iT) \\ &= p + \sum_{i:|N_i|=2} y(iT) - \sum_{i=1}^{p} y(iZ) \\ &= p + \sum_{i:|N_i|=2} y(iT) \ + \ y(ZT) \\ &= p + (s - p) = s. \end{aligned}$$

Therefore, (2) is satisfied. So we have feasible solution values $x_k(i)$ to the linear integer program that defines $g_s(\mathcal{N})$. The objective value of this solution is

$$\begin{aligned}
&\sum_{i=1}^{p}\sum_{k=1}^{|N_i|} f_k(N_i)x_k(i)\\
&\quad=\sum_{i=1}^{p} f_1(N_i)x_1(i)+\sum_{i:|N_i|=2} f_2(N_i)x_2(i)\\
&\quad=\sum_{i:|N_i|=2} f_1(N_i)(1-y(iT)-y(iZ))+\sum_{i:|N_i|=1} f_1(N_i)(1-y(iZ))\\
&\qquad\quad+\sum_{i:|N_i|=2} f_2(N_i)y(iT)\\
&\quad=\sum_{i=1}^{p} f_1(N_i)+\sum_{i=1}^{p}-f_1(N_i)y(iZ)+\sum_{i:|N_i|=2}(f_2(N_i)-f_1(N_i))y(iT)\\
&\quad=\sum_{i=1}^{p} f_1(N_i)+\sum_{i=1}^{p} w(iZ)y(iZ)+\sum_{i:|N_i|=2} w(iT)y(iT)\\
&\quad=\mu_s(\mathcal{N}).
\end{aligned}$$

So we conclude that $g_s(\mathcal{N})\geq\mu_s(\mathcal{N})$. Now, all of these steps are reversible (in particular, the affine map by which we construct x from y is invertible), so the result follows. □

3. Comparison with another matching bound

Hoffman, Lee and Williams [6, Section 1] considered a different bound based on matching. Let N^2 be the set of all subsets B of N satisfying $|B|=2$. For all $B\subset N^2$, we define binary variables $x(B)$. For convenience, we assume that s is even; for the odd case, we refer the interested reader to [6]. The *matching bound* of [6] is

$$\begin{aligned}
\psi_s^2=\ &\max \sum_{B\in N^2}(\ln\det C[B,B])\,x(B)\\
&\text{subject to}\sum_{B\in N^2} x(B)=s/2\ ;\\
&\sum_{\substack{B\in N^2:\\ j\in B}} x(B)\leq 1\ ,\forall\ j\in N\ ;\\
&x(B)\in\{0,1\},\ \forall\ B\in N^2\ .
\end{aligned}$$

By examples, we will demonstrate that the matching bound ψ_s^2 and our linear integer programming bound $g_s(\mathcal{N})$ with $|N_i|\leq 2$ are not generally comparable.

Example 1. Consider the matrix

$$C = \left(\begin{array}{cc|cc|c} \epsilon & 1 & 1 & 1 & 0 \\ 1 & \epsilon & 1 & 1 & 0 \\ \hline 1 & 1 & \delta & 1 & 0 \\ 1 & 1 & 1 & \delta & 0 \\ \hline 0 & 0 & 0 & 0 & \tau \end{array}\right),$$

where $n = 5$, the constants $\delta \neq \epsilon$ are greater than 1, $\tau > 0$ is sufficiently small, and we take $s = 4$. It is easy to check that $\psi_s^2 = \ln(\epsilon\delta - 1)^2$, which exceeds $g_s(\{1,2\},\{3,4\},\{5\}) = \ln((\epsilon^2 - 1)(\delta^2 - 1))$. So, the linear integer programming bound $g_s(\mathcal{N})$ with $|N_i| \leq 2$ can outperform the matching bound ψ_s^2.

Example 2. Consider the matrix

$$C = \left(\begin{array}{cc|c} \epsilon & 1 & 1 \\ 1 & \epsilon & 1 \\ \hline 1 & 1 & \delta \end{array}\right) = \left(\begin{array}{c|cc} \epsilon & 1 & 1 \\ \hline 1 & \epsilon & 1 \\ 1 & 1 & \delta \end{array}\right) = \left(\begin{array}{c|c|c} \epsilon & 1 & 1 \\ \hline 1 & \epsilon & 1 \\ \hline 1 & 1 & \delta \end{array}\right),$$

where $n = 3$, and we take $s = 2$, and $\delta > \epsilon > 1$. It is easy to check that $g_s(\mathcal{N}) = \ln(\epsilon\delta)$ for *all* partitions $\mathcal{N}$ of $N = \{1, 2, 3\}$ having the $|N_i| \leq 2$. Also, we have $\psi_s^2 = \ln(\epsilon\delta - 1)$. Therefore, there are examples for which the matching bound ψ_s^2 outperforms the linear integer programming bound $g_s(\mathcal{N})$ for *all* partitions having $|N_i| \leq 2$.

Empirical evidence collected by Hoffman, Lee and Williams [6, Table 2] suggests that, for data sets arising in practice, ψ_s^2 tends to be equal to or only slightly less than ψ_s^1. This, together with Proposition 2, suggests that for any partition having $|N_i| \leq 2$, the linear integer programming bound $g_s(\mathcal{N})$ should not do worse than the matching bound ψ_s^2. We can hope that finding a good partition $\mathcal{N}$ with $|N_i| \leq 2$ will enable the linear integer programming bound $g_s(\mathcal{N})$ to outperform the matching bound ψ_s^2 – this is borne out in our computational experiments (see Section 4).

4. Calculating a good partition and computational results

The value of the linear integer programming upper bound $g_s(\mathcal{N})$ depends on the choice of partition $\mathcal{N}$. As we will see, we can produce very good bounds, but we need a practical method for finding a good partition. We do not know the complexity of finding the best partition with $|N_i| \leq 2$, or any other interesting case. We suggest local search methods to find a good partition (see Hoffman, Lee and Williams [6, Figure 2]).

We experimented with our new bounds on the $n = 63$, $s = 31$ environmental-monitoring data set from [5]. Starting with the finest partition, we first calculated the diagonal bound ψ_s^1. Then, from this starting partition, we considered local-search steps involving moving a single element from its current block of the partition to another or a new block (these are the simplest moves considered in Hoffman, Lee and Williams [6]). In carrying out this search, we always took the best step available. But we did not always calculate the $f_k(N_i)$ exactly. We experimented with different values of Δ, and we only calculated

Table 1. Entropy gaps (Ohio Valley data: $n = 63$, $s = 31$)

INITIAL PARTITION:		$\psi_s^1 = 7.924975$	$\bar{\psi}_s^1 = 3.252440$
LOCAL SEARCH:	Δ	g_s	$\bar{g}_s$
	0	5.060646	2.629423
	1	3.705539	1.600504
	2	2.790565	1.235069
	3	1.961030	1.235069

the $f_k(N_i)$ exactly for $k \leq \Delta$ and $k \geq n - \Delta$. For $\Delta < k < n - \Delta$, we replaced $f_k(N_i)$ with the spectral upper bound $\sum_{l=1}^{k} \ln \lambda_l(N_i)$.

The resulting entropy gaps (i.e., bounds minus the optimal solution value) appear in Table 1. Note that the results for $\Delta = 0$, which amounts to local search with the spectral partition bound, appear in [6, Table 3, row 2a, columns 1c]. Our new results appear in the rows for $\Delta = 1, 2, 3$. The right-hand column uses the "complementary problem" (see the Introduction). We note that in [6, Table 3], the best gap that we had obtained for this problem was 2.521062; here, we have more than halved that!

The code that we used to test the *quality* of these bounds is only a prototype. An efficient implementation is required for successful use in a branch-and-bound solution algorithm. We expect to report on results with such an implementation in a forthcoming paper. Based on our earlier experience with branch-and-bound approaches to the maximum-entropy sampling problem (see [8, 9, 2]), we expect that our new ideas will have considerable positive impact on our ability to solve large problems.

Acknowledgements. Part of this research was carried out using the facilities of the High-Performance Computing Laboratory of the College of Arts and Sciences at the University of Kentucky, supported in part by NSF Grant DMS-9508543.

References

1. Kurt M. Anstreicher, Marcia Fampa, Jon Lee, and Joy Williams. Continuous relaxations for constrained maximum-entropy sampling. In *Integer Programming and Combinatorial Optimization (Vancouver, BC, 1996)*, volume 1084 of *Lecture Notes in Computer Science*, pages 234–248. Springer, Berlin, 1996.
2. Kurt M. Anstreicher, Marcia Fampa, Jon Lee, and Joy Williams. Using continuous nonlinear relaxations to solve constrained maximum-entropy sampling problems. *Mathematical Programming, Series A*, 85(2):221–240, 1999.
3. Jack Edmonds and Ellis L. Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial Structures and their Applications (Proc. Calgary Internat., Calgary, Alta., 1969)*, pages 89–92. Gordon and Breach, New York, 1970.
4. Ernst S. Fischer. Über den Hadamardschen determinantensatz. *Archiv für Mathematik und Physik*, 13(3):32–40, 1908.
5. Peter Guttorp, Nhu D. Le, Paul D. Sampson, and James V. Zidek. Using entropy in the redesign of an environmental monitoring network. In *Multivariate Environmental Statistics*, pages 175–202. North-Holland, Amsterdam, 1993.
6. Alan Hoffman, Jon Lee, and Joy Williams. New upper bounds for maximum-entropy sampling. In A.C. Atkinson, P. Hackl, and W.G. Müller, editors, *MODA 6 — Advances in model-oriented design and analysis*, pages 143–153. Springer-Verlag, 2001.
7. Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.
8. Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

9. Jon Lee. Constrained maximum-entropy sampling. *Operations Research*, 46(5):655–664, 1998.
10. Jon Lee. Semidefinite-programming in experimental design. In Henry Wolkowicz, Romesh Saigal and Lieven Vandenberghe, editors, *Handbook of Semidefinite Programming*, volume 27 of *International Series in Operations Research and Management Science*, pages 528–532. Kluwer, 2000.
11. Jon Lee. Maximum-entropy sampling. In Abdel H. El-Shaarawi and Walter W. Piegorsch, editors, *Encyclopedia of Environmetrics*, volume 3, pages 1229–1234. John Wiley & Sons Inc., 2001.
12. Alexander Schrijver. Min-max results in combinatorial optimization. In *Mathematical Programming: The State of the Art (Bonn, 1982)*, pages 439–500. Springer, Berlin, 1983.