CSC2414: Discrepancy Theory in Computer Science Fall 2015

Lecture 10 — November 30, 2015

Aleksandar Nikolov *Scribe: Lalla Mouatadid*

1 Approximation algorithms via discrepancy

In this lecture, we explore some applications of discrepancy to rounding and approximation algorithms. In particular, we look at two examples where by using discrepancy we obtain constant factor approximations to two well studied problems: scheduling on unrelated machines, and bin packing. But first, let us recall some definitions we will use later. Given a matrix $A \in \mathbb{R}^{m\times n}$, for the purposes of this lecture we will define linear discrepancy of A in a way that is slightly different from what we have seen before:

$$\begin{aligned}\operatorname{lindisc}(A) &= \max_{x\in[0,1]^n} \min_{y\in\{0,1\}^n} \|Ax - Ay\|_\infty \\ \operatorname{disc}(A) &= 2 \min_{y\in\{0,1\}^n} \|A((1/2)\cdot \mathbf{1} - Ay)\|_\infty\end{aligned}$$

Above, and in the rest of this lecture, $\mathbf{1}$ is the all-ones vector. The definition of linear discrepancy above differs from the one we are used to by a factor of 2.

By the theorem of Lovasz, Spencer, and Vesztergombi [5] that we saw in a previous lecture, we have the following

Theorem 1. $\forall A \in \mathbb{R}^{m\times n}:\ \operatorname{lindisc}(A) \le \operatorname{herdisc}(A)$

For $\{0,1\}$ matrices, i.e. incidence matrices of set systems, it is open whether we can reverse this inequality up to constant, or even logarithmic factors.

Linear discrepancy in fact allows us to round any real vactor, and not just vectors with entries in $[0,1]$. This is captures by the following proposition:

Proposition 2. $\forall x \in \mathbb{R}^n_+, \exists y \in \mathbb{Z}^n_+$ *such that* $\|Ax - Ay\|_\infty \le \operatorname{lindisc}(A)$

Proof. Let $\bar{x}_i = x_i - \lfloor x_i \rfloor$, then $\bar{x}_i \in [0,1]$, and there exists $\bar{y} \in \{0,1\}^n$ and $y = \bar{y} + \lfloor x \rfloor$ (where the floor is applied coordinate-wise) such that:

$$\begin{aligned}\|A\bar{x} - A\bar{y}\|_\infty &\le \operatorname{lindisc}(A) \\ \|(A\bar{x} + A\lfloor x\rfloor) - (A\bar{y} + A\lfloor x\rfloor)\|_\infty &\le \operatorname{lindisc}(A) \\ \|Ax - Ay\|_\infty &\le \operatorname{lindisc}(A)\end{aligned}$$

□

2 Scheduling on unrelated parallel machines

Consider the following setting: Given m machines and n jobs, let p_{ij} denote the time job j takes on machine i. The goal is to assign each job to a machine so all machines are done processing by time T. Each machine executes only one job at a time. The time T by which the machines are done processing is called the makespan. There exists a 2-approximation algorithm by Lenstra, Shmoys and Tardos [4], and it is open whether the approximation can be improved to a factor 1.5, which would be optimal, unless P=NP.

In this section, we will use discrepancy to give a constant factor approximation to this problem, albeit not as good as the best known factor. Consider the following linear program, where we guess the optimum makespan T by binary search, and drop the jobs that have processing time greater than T.

$$\begin{aligned}
&\sum_{i=1}^{m} x_{ij} = 1 && \forall \text{ jobs } j \in [n] \\
&\sum_{j=1}^{n} \frac{p_{ij}}{T} x_{ij} \leq T && \forall \text{ machines } i \in [m] \\
&x_{ij} = 0 && \forall i, j \text{ s.t. } p_{ij} > T \\
&0 \leq x_{ij} \leq 1
\end{aligned}$$

Fix x to be an x that solves this feasibility problem, and consider the constraint matrix $A \in [0,1]^{(m+n)\times|E|}$ where $E = \{(i,j) : p_{ij} \leq T\}$. (The fact that all entries of A are in $[0,1]$ is guaranteed because we drop all (i,j) such that $p_{ij} > T$.) Let $D = \text{herdisc}(A)$, and define $\bar{x} = (1+D)x$. We will be rounding this scaled-up solution, rather than x. Intuitively, we are over-scheduling the jobs (scheduling each job multiple times), so that even after rounding we know that each job will be scheduled at least once. You can think of this step as creating D extra copies of each job, so that after rounding at least one of the copies is scheduled.

Let $y \in \mathbb{Z}_+^n$ be such that

$$\begin{aligned}
\|A\bar{x} - Ay\|_\infty &\leq \text{lindisc}(A) \\
&\leq \text{herdisc}(A) && \text{by Proposition 2.}
\end{aligned}$$

Unpacking this inequality, from the job-constraints of the LP we have for each $j \in [n]$:

$$\begin{aligned}
\left| \sum_{i=1}^{m} \bar{x}_{ij} - \sum_{i=1}^{m} y_{ij} \right| &\leq D \\
\sum_{i=1}^{m} y_{ij} &\geq \sum_{i=1}^{m} \bar{x}_{ij} - D \\
&\geq (1+D) - D = 1
\end{aligned}$$

This implies that if we treat y as an assignment of jobs to machines, then each job is assigned to at least one machine. It could happen that $\sum_{i=1}^{m} y_{ij} > 1$, in which case a job is assigned to more than one machine. If that happens, pick an arbitrary machine j such that $y_{ij} > 0$ to assign job i to: dropping all other assignments only decreases the makespan.

From the machine constraint of the LP, for each $i \in [m]$ we have:

$$\left|\frac{1}{T}\sum_{j=1}^{n}\bar{x}_{ij}p_{ij} - \frac{1}{T}\sum_{j=1}^{n}y_{ij}p_{ij}\right| \leq D$$

$$\frac{1}{T}\sum_{j=1}^{n}\bar{x}_{ij}p_{ij} \leq 1 + D \implies \frac{1}{T}\sum_{j=1}^{n}y_{ij}p_{ij} \leq (1+D) + D$$

$$\implies \sum_{j=1}^{n}y_{ij}p_{ij} \leq (1+2D)T$$

Therefore, the makespan of the assignment given by y is at most $(1+2D)T$.

Notice that by construction, each column in the matrix A has 2 non-zero entries, each bounded by 1 in absolute value. Using an easy modification of the Beck-Fiala theorem, we know that $D = \text{herdisc}(A) \leq 2\Delta$ where Δ is the maximum number of non-zero entries in any column of A. Thus $D \leq 4$, and the makespan of the assignment given by y is at most $9T$. This implies a factor 9 approximation for minimizing makespan on unrelated parallel machines.

3 Bin Packing

We next consider a second, more sophisticated application of of discrepancy to approximation algorithms. The bin packing problem takes as input a set of n items of sizes $s_1, s_2, \ldots, s_n$, where each $s_i \in [0,1]$. The goal is to pack the items into the smallest number of bins, each of size 1. Bin packing is NP-hard, and remains NP-hard to decide whether we need 2 vs. 3 bins (i.e. whether OPT ≤ 2 or ≥ 3). Before getting started, we list the current known approximations:

- The First-Fit (arbitrary order) algorithm gives a 2-approximation.
- The First-Fit in decreasing order gives a 1.7 OPT + 1 [2].
- De La Vega and Lucker gave an asymptotic PTAS that uses $\leq (1+\epsilon)\text{OPT} + 1$ bins, for any given $\epsilon \geq 0$.
- Karmarkar and Karp (KK) gave an algorithm that achieves $\leq \text{OPT} + O(\log^2(\text{OPT}))$ many bins [3].
- Rothvoss used discrepancy to give an algorithm that uses $\leq \text{OPT} + O(\log(\text{OPT})\log\log(OPT))$ bins [6].
- Rothvoss and Hoberg refined the discrepancy approach to give an algorithm that uses $\leq \text{OPT} + O(\log(\text{OPT}))$ bins [1].

For the special case when $s_i > \frac{1}{k+1}, \forall i$ (i.e. we can place at most k items per bin), and k is a constant, the KK result already gives $\text{OPT} + O(\log(\text{OPT}))$ bins. Hoberg and Rothvoss's result essentially reduces the general case to this case. It is open whether there exists an efficient algorithm that packs all items in OPT + 1 bins (!).

For the remainder of the lecture, we will show how to get the KK bound for $s_i > \frac{1}{4}, \forall i$ using discrepancy on the Gilmore Gomory LP relaxation, defined below. For convenience, let us assume all the weights are sorted in decreasing order:

$$1 \geq s_1 \geq s_2 \geq \ldots \geq s_n > \frac{1}{4}$$

Let $s = (s_1, s_2, \ldots, s_n)$ denote the sorted vector, and let $\mathcal{P}_s$ be all possible ways to pack the items:

$$\mathcal{P}_s = \left\{ p \in \{0,1\}^n \;\; : \;\; p^T s = \sum p_i s_i \leq 1 \right\}$$

Think of p as the indicator of a feasible set of items which we can assign to a single bin. Since we can only fit at most 3 items per bin, it follows that $|\mathcal{P}_s| = O(n^3)$. Now consider the following linear program, which is a relaxation of the bin packing problem:

$$\begin{array}{lll} \text{minimize} & \sum x_p & (\#\text{ of bins}) \\ \text{subject to} & \sum x_p \cdot p \geq 1 & (\text{each item in at least a bin}) \\ & x_p \geq 0 \;\; \forall p \in \mathcal{P}_s & \end{array}$$

Let x be an optimal x such that $|\{p : x_p > 0\}| \leq n$. That such an optimal solution always exists follows from the theory of basic feasible solutions. Let B be the constraint matrix of the LP, with columns restricted to p such that $x_p > 0$, and let us replace x with x restricted to these patterns p, as well. Given the item size constraint, notice that each column in B has at most 3 zeroes:

$$B = \begin{pmatrix} \vdots & & \vdots \\ p_1 & \ldots & p_n \\ \vdots & & \vdots \end{pmatrix} \qquad \{p_1 \ldots p_n\} = \{p : x_p > 0\}$$

$$Bx = 1$$

Let A be a matrix constructed as follows:

$$A = \begin{pmatrix} T_n B \\ 3 \ldots 3 \end{pmatrix}$$

where T_n is the lower triangular $\{0,1\}$ matrix:

$$\begin{pmatrix} 1 & \ldots & 0 \\ 1 & \ddots & 0 \\ 1 & \ldots & 1 \end{pmatrix}$$

This means that each of the first n rows of A is of the form $A_{i*} = \sum_{j=1}^{i} B_{j*}$. Since the columns of B have at most 3 ones each, and all other entries are 0, it follows that $A \in \{0,1,2,3\}^{(n+1)\times n}$. Moreover, each column in A is monotone non-decreasing. Because $Bx = 1$, we have:

$$Ax = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n \\ 3 \cdot LP \end{pmatrix}$$

where $LP \leq OPT$. Assume $x_p \in [0,1)$, and let D denote lindisc(A). By Proposition 2,

$$\begin{aligned} \exists y \in \mathbb{Z}_+^n \text{ s.t. } \|Ax - Ay\|_\infty &\leq D \\ |3 \cdot \mathbf{1}^T y - 3 \cdot \mathbf{1}^T x| &\leq D \\ 1^T y \leq \text{ LP } &- \frac{1}{3} D \end{aligned}$$

Claim 3. *We can pack all items in $D + \sum_p y_p$ many bins.*

Proof. We will use y_p copies of pattern p, and additional D copies of the pattern that contains a space for item 1. We will pack items according to thse patterns, except that we will also allow ourselves to put a smaller item in the space reserved by a larger item. Since item 1 is the largest one[1], we can pack any item in a space reserved for it.

We construct a bipartite graph $G(V = \mathcal{U} \cup \mathcal{V}, E)$ as follows: Every vertex $u_i \in \mathcal{U}$ corresponds to an item i of size s_i. For item 1, we create $(By)_1 + D$ copies of v_1, which is the number of slots where item 1 can fit (or any item with size $\leq s_1$). For the remaining items, create $(By)_i$ copies of vertex v_i for each item i (i.e. the # of slots for items of size $\leq s_i$). For every $u_i \in \mathcal{U}$, add an edge $(u_i, v_j) \in E$ for all $j \leq i$. The goal is to compute a $\mathcal{U}$-perfect matching. Then, for any edge (u_i, v_j) of the matching, we will pack item i in any slot reserved for item j. Since the matching is perfect, we can pack all items.

The proof that a $\mathcal{U}$-perfect matching exists is based on Hall's theorem, which says:

Theorem 4. *$\mathcal{U}$-perfect matching exists if and only if for all $\mathcal{W} \subseteq \mathcal{U}, |N(\mathcal{W})| \geq |\mathcal{W}|$, where $N(\mathcal{W})$ is the set of neighbors of $\mathcal{W}$.*

Now we verify the condition of Hall's theorem. Assume $\mathcal{W} = \{u_1, \ldots, u_i\}$. This is without loss of generality, because $\forall j \leq i$, $N(u_j) \subseteq N(u_i)$ by construction. Therefore

$$\begin{aligned} |N(\mathcal{W})| &= \sum_{j=1}^{i} (By)_j + D \quad \text{for } j = 1 \\ &= (Ay)_i + D \\ &\geq (Ax)_i = i = |\mathcal{W}| \end{aligned}$$

[1] recall that items are sorted in decreasing order

Therefore a $\mathcal{U}$-perfect matching exists and all items can be packed, and the cost of the solution is at most $\leq$ LP $+(1+\frac{1}{3})D$. □

On the other hand, for all A with monotone non-decreasing columns and entries in $[k]$, we have $\text{herdisc}(A) = O(k \log n)$, thus:

$$\begin{aligned} D &= \text{lindisc}(A) \\ &\leq \text{herdisc}(A) \\ &= O(3 \log n) = O(\log n) \end{aligned}$$

We will not prove the bound on the hereditary discrepancy of A here: there is a clever proof which uses techniques similar to these in the proof of the Beck-Fiala theorem. Let us, however, analyze the γ_2 norm instead:

Proposition 5. $\gamma_2(A) = O(\log n)$

Proof. We decompose A into 3 matrices where the entries are of the form:

$$A_{ij}^{(k)} = \begin{cases} 1 \text{ if } A_{ij} \geq k \\ 0 \text{ otherwise} \end{cases}$$

Therefore $A = A^{(1)} + A^{(2)} + A^{(3)}$. Since $\gamma_2(A)$ satisfies the triangle inequality,

$$\gamma_2(A) \leq \sum_{k=1}^{3} \gamma_2(A^{(k)})$$

A has monotone columns, thus each $A^{(k)}$ has monotone columns. Up to columns rearrangement, $A^{(k)}$ is a submatrix of T_n, and, since γ_2 does not change when considering submatrices and/or rearranging columns, it follows that:

$$\gamma_2(A^{(k)}) \leq \gamma_2(T_n) = \Theta(\log n)$$

□

Now using Proposition 5 and the bound on hereditary discrepancy in terms of the γ_2 norm that we proved in a prior lecture, we get

$$\text{herdisc}(A) = O(\log^{\frac{3}{2}} n).$$

As we mentioned above, a specialized argument, in the spirit of the proof of the Beck-Fiala theorem, shows that $\text{herdisc}(A) = O(\log n)$.

References

[1] Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. *CoRR*, abs/1503.08796, 2015.

[2] David S Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

[3] Narendra Karmarkar and Richard M Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 312–320. IEEE, 1982.

[4] Jan Karel Lenstra, David B Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1-3):259–271, 1990.

[5] László Lovász, Joel Spencer, and Katalin Vesztergombi. Discrepancy of set-systems and matrices. *European Journal of Combinatorics*, 7(2):151–160, 1986.

[6] Thomas Rothvoß. Approximating bin packing within o(log OPT * log log OPT) bins. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 20–29. IEEE Computer Society, 2013.