# CS364B: Frontiers in Mechanism Design
# Lecture #10: Coverage Valuations and Convex Rounding*

Tim Roughgarden†

February 5, 2014

## 1 Coverage Valuations

Recall the setting of submodular bidder valuations, introduced in Lecture #7.

**Scenario #6:**

- A set $U$ of $m$ non-identical items.
- Each bidder $i$ has a private valuation $v_i : 2^U \to \mathcal{R}^+$ that is *submodular*, meaning that for every pair of sets $S \subseteq T \subseteq U$ and item $j$,

$$v_i(T \cup \{j\}) - v_i(T) \leq v_i(S \cup \{j\}) - v_i(S). \tag{1}$$

Recall that the submodularity condition in (1) is a form of diminishing returns, but is strictly weaker than the gross substitute condition. If we ignore incentives and assume straightforward bidding, the Kelso-Crawford auction achieves a $\frac{1}{2}$-approximation of the optimal welfare. Another $\frac{1}{2}$-approximation is given in Exercise #25. Are there DSIC mechanisms with equally good guarantees?

While the answer is "no" in general (see Section 7), this lecture gives an affirmative answer for an interesting subclass of submodular valuations: *coverage* valuations.

**Definition 1.1** A *coverage valuation function* $v_i : 2^U \to \mathcal{R}^+$ has the form

$$v_i(S) = \left| \bigcup_{j \in S} A_{ij} \right|,$$

where $A_{i1}, \ldots, A_{im}$ are subsets of a ground set $X_i$.



†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

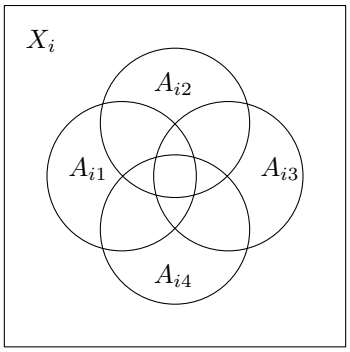


Figure 1: A coverage valuation function with subsets $A_{ij}$ of a ground set $X_i$.

That is, the valuation is the "area" covered by the sets that correspond to the items in $S$. See Figure 1.

For example, items might be broadcasting licenses, $X_i$ might be a set of cities, and $A_{ij}$ the cities for which the $j$th license grants broadcasting rights. For another example, $X_i$ might denote a set of skills relevant to the firm $i$, and $A_{ij}$ the set of skills possessed by the $j$th worker. In both cases, the valuation is simply the number of cities or skills covered by a collection of items.

Every coverage valuation is submodular — the number of elements newly covered by another set is decreasing in the number of current sets — and there are coverage valuations that do not satisfy the GS condition (see Exercises). Not every GS valuation can be represented as a coverage valuation (see Exercises), although the main results of this lecture do apply to all GS valuations, and convex combinations thereof [5].

Throughout this lecture, we assume that each valuation $v_i$ is a coverage valuation, given by an explicit list of the $A_{ij}$'s, with each $A_{ij}$ represented as a list of elements (from $X_i$). The goal is to design a DSIC mechanism that runs in time polynomial in this input size and has near-optimal welfare.

Even ignoring incentive issues, maximizing the welfare with explicitly given coverage valuations is a hard problem.

**Theorem 1.2 ([8])** *Assuming $P \neq NP$, for every $\epsilon > 0$, there is no polynomial-time $(1 - \frac{1}{e} + \epsilon)$-approximation algorithm for maximizing the welfare when all bidders have explicitly described coverage valuations.*

Note that $1 - \frac{1}{e} \approx 0.63$. The proof of Theorem 1.2 is a non-trivial reduction from 3-coloring, and we won't cover it here. Without incentive constraints, there are $(1 - \frac{1}{e})$-approximation algorithms [4, 9] for the problem. The main goal of this lecture is to achieve an equally good guarantee via a DSIC mechanism.

**Theorem 1.3 ([5])** *For the subclass of Scenario #6 with explicitly described coverage valuations, there is a MIDR allocation rule (and hence a DSIC mechanism) that runs in expected polynomial time and computes an allocation with expected welfare at least $1 - \frac{1}{e}$ times the maximum possible.*

## 2 Why Scaling Algorithms Aren't Enough

Last lecture introduced a general technique for converting approximation algorithms into equally good MIDR allocation rules and hence DSIC mechanisms. Since we already have $(1-\frac{1}{e})$-approximation algorithms for maximizing welfare with known coverage valuations [4, 9], why can't we apply that technique here, "smoothing out" the error to obtain equally good scaling algorithms?

There are a couple of serious obstacles. One technical point is that the scaling algorithm-based technique from last lecture worked with an optimal solution to the linear programming relaxation of the welfare-maximization problem, and solving this linear program in polynomial time requires efficiently computable demand queries for the ellipsoid method's separation oracle. Answering a demand query for an explicitly given coverage valuation is an $NP$-hard problem (see Exercises). (Value queries, by contrast, are easy.) Thus, it's not clear how to get this approach off of the ground.

There is also a more fundamental problem with the scaling algorithm-based approach. Recall the point in last lecture where actually invoked our $\alpha$-approximation algorithm. We formulated a linear system (LP2) for which solutions correspond to $\alpha$-scaling algorithms, and solved the dual linear system (D2) using the ellipsoid method. The separation oracle for this invocation of the ellipsoid method boiled down to the following problem: given "pseudo-valuations" $z_i(S)$ for certain bidder-bundle pairs $(i, S) \in I$ and a fractional solution $\mathbf{y}^*$, produce an integer solution with pseudo-welfare at least $\alpha$ times $\sum_{(i,S)\in I} z_i(S)y^*_{iS}$. Recall that the pseudo-valuations $\mathbf{z}$, as some candidate solution to (D2) that is opaquely generated by the ellipsoid method, *have no obvious structure whatsoever* — they can be negative, non-monotone, etc. Last lecture we overcame this obstacle because, when there is at a logarithmic number of copies of every good, there is a good approximation algorithm even with arbitrary valuations. With only one copy per good and general valuations, there are no good approximation algorithm ($\Theta(\sqrt{m})$ is achievable and best-possible).

In Scenario #6 and special cases of it, the only reason we might be able to achieve a good approximation factor is because of the structure in the valuations. Unfortunately, structure in the valuations $\mathbf{v}$ need not translate to analogous structure in the pseudo-valuations $\mathbf{z}$ generated by the ellipsoid method when we solve the dual linear system (D2). For this reason, it is not clear that the existence of an $\alpha$-approximation algorithm for a particular valuation class implies the existence of an $\alpha$-scaling algorithm for the same valuation class.

## 3 Convex Rounding

Last lecture proposed the idea of converting approximation algorithms based on randomized rounding into equally good MIDR allocation rules, and introduced scaling algorithms as a subclass of rounding algorithms that naturally lead to such rules. Can we go beyond scaling algorithms?

Let's revisit our visualization of a randomized rounding approximation algorithm as the composition of a relaxation algorithm and an oblivious rounding algorithm (Figure 2). The

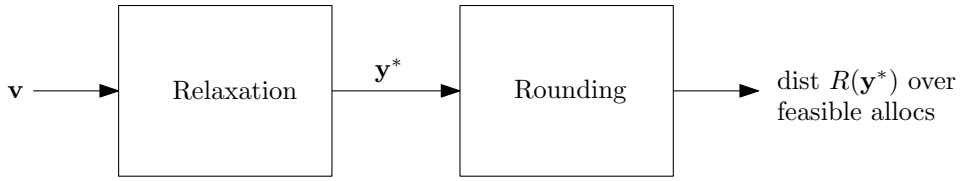


Figure 2:

relaxation algorithm takes as input a valuation profile $\mathbf{v}$ (e.g., explicitly described coverage valuations) and outputs an optimal solution $\mathbf{y}^*$ to a linear programming relaxation, and the rounding algorithm compiles the fractional solution $\mathbf{y}^*$ into a distribution $R(\mathbf{y}^*)$ over integer solutions.

Recall from last lecture the challenge in converting randomized rounding algorithms into MIDR allocation rules: the relaxation algorithm optimizes over fractional solutions $\mathbf{y}^*$, while the MIDR condition requires optimizing over the consequent distribution $R(\mathbf{y}^*)$. By design, scaling algorithms make these two optimization problems one and the same. It's not clear what other rounding algorithms could possibly have this property, since non-scaling rounding algorithms disconnect the choice of output distribution from the optimization step.

Here's a crazy idea for obtaining the MIDR condition with a non-scaling algorithm $R$: optimize directly over the *output* $R(\mathbf{y})$ of the rounding algorithm, rather than over the *input* $\mathbf{y}$. That is, why not just directly solve the optimization problem:

$$\max_{\mathbf{y}} \mathbf{E}_{\omega \sim R(\mathbf{y})} \left[ \sum_{i=1}^{n} v_i(\omega) \right] \tag{2}$$

$$\text{subject to } \mathbf{y} \text{ feasible for some linear system.} \tag{3}$$

In effect, this formulation looks ahead to the result of the rounding algorithm $R$ when performing its optimization.

The obvious benefit of this approach is that it yields an allocation rule that is MIDR by construction. The range of the allocation rule is some subset of $\{R(\mathbf{y}) : \mathbf{y} \text{ feasible}\}$. For every valuation profile $\mathbf{v}$, by definition the rule selects the distribution of $\{R(\mathbf{y}) : \mathbf{y} \text{ feasible}\}$ that maximizes the expected welfare. The obvious drawback is that the objective function (2) looks painful to optimize for non-trivial rounding algorithms $R$.

For the LP relaxation from Lecture #6 and an $\alpha$-scaling algorithm $R$, linearity of expectation implies that the objective function (2) simplifies to the linear (in $\mathbf{y}$) objective

$$\max_{\mathbf{y}} \alpha \sum_{i=1}^{n} \sum_{S \subseteq U} v_i(S) y_{iS}.$$

That is, since looking ahead to the output $R(\mathbf{y})$ of an $\alpha$-scaling algorithm $R$ just scales the welfare of $\mathbf{y}$ by an $\alpha$ factor, the objective function (2) is linear and can be optimized over efficiently. Conversely, if the goal is a linear objective function, there aren't a lot of options beyond scaling algorithms.

More generally, concave objective functions can be maximized over a linear system in polynomial time, for example via the ellipsoid method (modulo technical details, see Appendix A). By a *convex rounding algorithm*, we mean a rounding algorithm $R$ such the objective function (2) is concave. The perhaps implausible hope is that there are non-scaling convex rounding algorithms that guarantee a good approximation for problems that we care about. Such a rounding algorithm would yield a polynomial-time MIDR allocation rule with a good approximation guarantee.

# 4 A Non-Example

This section gives a failed attempt at designing a convex rounding algorithm for the special case of Scenario #6 with coverage valuations. This exercise will lead us naturally to a related rounding algorithm that is indeed convex.

For the rest of this lecture, we use the following simple set of linear feasibility constraints, which only keeps track of item assignments, rather than bundle assignments:

$$\begin{aligned} \sum_{i=1}^{n} y_{ij} &\leq 1 \qquad \text{for all items } j \\ y_{ij} &\geq 0 \qquad \text{for all bidders } i \text{ and items } j. \end{aligned}$$

This linear system is similar to the one we used to model unit-demand bidders (Lecture #2), but without the constraints of the form $\sum_{j \in U} y_{ij} \leq 1$. One motivation for using this relatively simple feasible region is that, to prove concavity of an objective function of the form

$$\max_{\mathbf{y}} \mathbf{E}_{\omega \sim R(\mathbf{y})} \left[ \sum_{i=1}^{n} v_i(\omega) \right],$$

it helps to have a small and simple set of decision variables.

The simplest rounding algorithm $R$ imaginable is to assign independently each good $j$ to a random bidder $i$, with bidder $i$ receiving item $j$ with probability $y_{ij}$. Let's try to prove that corresponding objective function $\mathbf{E}_{\omega \sim R(\mathbf{y})}[\sum_{i=1}^{n} v_i(\omega)]$ is concave.

Before we get started, we should come clean and point out that *this could never work*. Why? Suppose that $\mathbf{y}$ is an integral (i.e., 0-1) solution. Then, by the definition of $R$, $R(\mathbf{y}) = \mathbf{y}$ with probability 1. Thus, the range $\{R(\mathbf{y}) : \mathbf{y} \text{ feasible}\}$ includes all feasible integral points, and the optimization problem (2)–(3) is equivalent to the original welfare-maximization problem. (The expected welfare of a distribution over allocations is at most the maximum welfare of a feasible allocation, so in this case the optimal solution to (2)–(3) is always a integral solution.)

Still, it will be very useful to see exactly how our doomed attempt to prove concavity breaks down. Fix a profile $\mathbf{v}$ of coverage valuations. Since the sum of concave functions is again concave, we can prove that the function (2) is concave in $\mathbf{y}$ by proving that

$$\mathbf{E}_{S_i \sim R(\mathbf{y})}[v_i(S_i)] \tag{4}$$

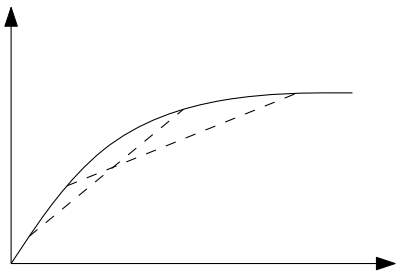

Figure 3: Concave functions have all chords under the function.

is concave in $\mathbf{y}$ for each bidder $i$. (Abusing notation, "$S_i \sim R(\mathbf{y})$" denotes the random bundle given to bidder $i$ by the rounding algorithm $R$ with fractional solution $\mathbf{y}$.)

Next, since $v_i$ is a coverage function, it has the form

$$v_i(S_i) = \left| \bigcup_{j \in S_i} A_{ij} \right|,$$

where $A_{i1}, \ldots, A_{im}$ are subsets of a ground set $X_i$. Using linearity of expectations, we can write

$$\mathbf{E}_{S_i \sim R(\mathbf{y})}[v_i(S_i)] = \sum_{a \in X_i} \mathbf{Pr}_{R(\mathbf{y})}[a \in \cup_{j \in S_i} A_{ij}] \,.$$

Using again that the sum of concave functions is concave, to prove that (4) is concave we only need to show that

$$\mathbf{Pr}_{R(\mathbf{y})}[a \in \cup_{j \in S_i} A_{ij}]$$

is concave in $\mathbf{y}$ for every $a \in X_i$. The good news is that this is a very simple expression that we can analyze in detail. The bad news is that

$$\mathbf{Pr}_{R(\mathbf{y})}[a \in \cup_{j \in S_i} A_{ij}] = 1 - \prod_{j \,:\, a \in A_{ij}} (1 - y_{ij}), \tag{5}$$

which is not concave in $\mathbf{y}$. To see this, recall that concave functions are characterized by having all chords lying below the graph (Figure 3). In particular, $f(\frac{1}{2}\mathbf{y}^1 + \frac{1}{2}\mathbf{y}^2) \geq \frac{1}{2}f(\mathbf{y}^1) + \frac{1}{2}f(\mathbf{y}^2)$. By contrast, the function $f$ in (5) satisfies $f(\mathbf{y}) = 1$ for every $\mathbf{y}$ with a "1" in one component and 0 in all the rest, while $f(\frac{1}{2}\mathbf{y}^1 + \frac{1}{2}\mathbf{y}^2) = \frac{3}{4}$ for two such solutions $\mathbf{y}^1, \mathbf{y}^2$ with "1"s in different components.

# 5 A Convex Rounding Algorithm

Let's try to salvage the failed attempt of Section 4. The simplest-possible tweak to that rounding algorithm would be to assign each item $j$ independently according to probabilities $\{f(y_{ij})\}_{i=1}^n$ for some function $f$. (Last section, $f(x) = x$.) What properties of $f$ do we want?

1. Recall our observation that if $f(1) = 1$ then all integer feasible solutions belong to the range of $R(\mathbf{y})$ and so optimizing over the range of $R$ is as hard as the original welfare-maximization problem. So, we need $f(1) < 1$ to have any hope of optimizing efficiently over the range of $R$. More generally, the hardness result of Theorem 1.2 suggests that we'll need to take $f(1) \leq 1 - \frac{1}{e}$; see also the final derivation in Section 6 below. This clue that the best-case scenario is $f(1) = 1 - \frac{1}{e}$ is significant — it's good to know what you're shooting for!

2. Given that we're planning to assign an item with probabilities corresponding to the $f(y_{ij})$'s, we want $\sum_{i=1}^{n} f(y_{ij})$ for every $j$ in every $\mathbf{y}$. Since $\sum_{i=1}^{n} y_{ij} \leq 1$, an easy way to ensure this is to choose an $f$ satisfying $f(x) \leq x$ for all $x \in [0, 1]$.

3. We want the analysis of last lecture to go through. That is, we want the right-hand side of (5),
$$1 - \prod_{j\,:\,a \in A_{ij}} (1 - f(y_{ij})),$$
to be concave in $\mathbf{y}$. Equivalently, we want
$$\prod_{j\,:\,a \in A_{ij}} (1 - f(y_{ij})) \tag{6}$$
to be convex in $\mathbf{y}$.

Remarkably, the simplest-imaginable function $f$ that satisfies the first two properties also satisfies the third. Namely, take $f(x) = 1 - e^{-x}$. Then, $f(1) = 1 - \frac{1}{e}$ and $f(x) \leq x$ for all $x \in [0, 1]$ (Figure 4) — more generally, $f(x) \in [(1 - \frac{1}{e})x, x]$ for all $x \in [0, 1]$. The expression (6) evaluates to
$$\exp\{-\sum_{j\,:\,a \in A_{ij}} y_{ij}\}, \tag{7}$$
which is convex in $\mathbf{y}$![1]

Summarizing, here is the final rounding algorithm $R$: given $\mathbf{y}$, independently for each item $j$, award item $j$ to bidder $i$ with probability $1 - e^{-y_{ij}}$, and to no one with the remaining probability $1 - \sum_{i=1}^{n}(1 - e^{-y_{ij}})$, This is a well-defined rounding procedure because $\sum_{i=1}^{n} y_{ij} \leq 1$ for every $j$ and $f(y_{ij}) \leq y_{ij}$ for every $i$ and $j$. For an integer solution $\mathbf{y}$, $R(\mathbf{y})$ corresponds to the scaled-down version $(1 - \frac{1}{e})\mathbf{y}$.[2]

[1] The exponential function is convex. If you prefer a formal proof, consider the Hessian matrix of (6). It has a single block of non-zeros, corresponding to the items $j$ such that $a \in A_{ij}$, and all entries in this block have the same value $\exp\{-\sum_{j\,:\,a \in A_{ij}} y_{ij}\} > 0$. This is a rank-one, positive semidefinite matrix. Hence, the function in (6) is convex.

[2] The exact same rounding algorithm $R$ remains convex when each valuation $v_i$ is a convex combination of valuations that satisfy the gross substitutes (GS) condition. The reason is that, when $v_i$ is a GS valuation, the properties of such valuations identified in the Bonus Lecture imply that the Hessian matrix of $\mathbf{E}_{S_i \sim R(\mathbf{y})}[v_i(S_i)]$ is an affine combination of matrices that each consist of a single block of nonzero entries, all equal to a common negative scalar. Such a matrix is negative semidefinite. Negative semidefiniteness is preserved when taking convex combinations of GS valuations.

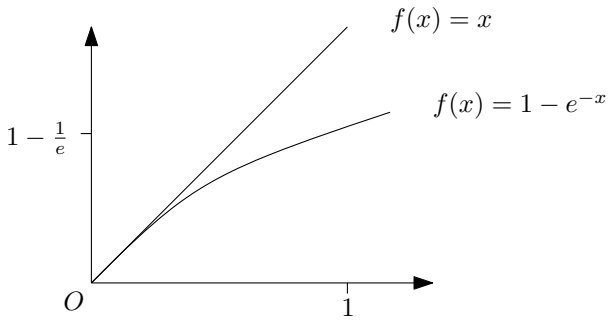


Figure 4:

# 6 Properties of the Convex Rounding Algorithm $R$

Let $\mathbf{x}$ be the allocation rule that optimally solves the optimization problem (2)–(3), where $R$ is the rounding algorithm from Section 5, and then uses $R$ to generate a random allocation from the optimal solution $\mathbf{y}^*$. First, as noted in Section 3, this rule is MIDR by construction. Second, for coverage valuations, the objective function

$$\max_{\mathbf{y}} \sum_{i=1}^{n} \sum_{a \in X_i} \left( 1 - \prod_{j \,:\, a \in A_{ij}} e^{-y_{ij}} \right), \tag{8}$$

is concave. Since the constraints (3) are linear, the allocation rule $\mathbf{x}$ can be implemented in polynomial time for such valuations. (See Appendix A for the technical fine print.) Finally, we prove below that, for every profile of submodular valuations $\mathbf{v}$, the rule $\mathbf{x}$ yields an outcome with expected welfare at least $1 - \frac{1}{e}$ times the maximum possible.

Fix a profile $\mathbf{v}$ of submodular valuations. We exhibit a feasible solution $\hat{\mathbf{y}}$ to (3) such that the expected welfare of $R(\hat{\mathbf{y}})$ is at least $1 - \frac{1}{e}$ times the maximum possible. Since $\mathbf{x}$ is MIDR, the expected welfare of its output is at least as large.

Let $(S_1^*, \ldots, S_n^*)$ denote a welfare-maximizing allocation. Define $\hat{y}_{ij} = 1$ if $j \in S_i^*$ and 0 otherwise. To lower bound the expected welfare of $R(\hat{\mathbf{y}})$, consider a bidder $i$ and relabel items so that $S_i^* = \{1, 2, \ldots, \ell\}$. Bidder $i$ receives each item $j = 1, 2, \ldots, \ell$ independently with probability $1 - \frac{1}{e}$. By linearity of expectation, we can write

$$\begin{aligned}
\mathbf{E}_{S_i \sim R(\hat{\mathbf{y}})}[v_i(S_i)] &= \sum_{j=1}^{\ell} \mathbf{Pr}_{R(\hat{\mathbf{y}})}[i \text{ gets } j] \, \mathbf{E}_{S_i \sim R(\hat{\mathbf{y}})}[v_i(S_i \cap \{1, 2, \ldots, j\}) - v_i(S_i \cap \{1, 2, \ldots, j-1\})] \\
&\geq \sum_{j=1}^{\ell} \mathbf{Pr}_{R(\hat{\mathbf{y}})}[i \text{ gets } j] \, (v_i(\{1, 2, \ldots, j\}) - v_i(\{1, 2, \ldots, j-1\}) \\
&= \left(1 - \frac{1}{e}\right) \cdot v_i(S_i^*),
\end{aligned}$$

where the inequality follows from the submodularity of $v_i$. Summing over the bidders and

using linearity of expectation proves that the expected welfare of the allocation output by $\mathbf{x}$ is at least $1 - \frac{1}{e}$ times the maximum possible, as claimed.

# 7 General Submodular Valuations

We began Part II of this course with a challenge question: given that there are good approximation algorithms for maximizing welfare with submodular bidder valuations, are there equally good DSIC approximation mechanisms? This lecture gave a positive result for some well-motivated subclasses of submodular valuations, but what about the general case?

We conclude Part II of the course by mentioning negative results for DSIC mechanisms for general submodular valuations (Scenario #6). Without any incentive constraints, there is a polynomial-time $(1 - \frac{1}{e})$-approximation algorithm for welfare maximization for arbitrary monotone submodular valuations that support value queries [9]. However, the only DSIC mechanisms with a non-trivial (i.e., sub-polynomial) approximation factor must use an exponential number of value queries in the worst case [3, 6]. Thus, incentive constraints make the optimization problem far harder. The proof of this gulf between the power of polynomial-time DSIC mechanisms and of general polynomial-time algorithms for combinatorial auctions is a highlight of recent theoretical work in algorithmic mechanism design.[3]

# References


[1] S. Dobzinski and J. Vondrak. Communication complexity of combinatorial auctions with submodular valuations. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1205–1215, 2013.

[2] Shahar Dobzinski. Two randomized mechanisms for combinatorial auctions. In *Proceedings of APPROX-RANDOM*, pages 89–103, 2007.

[3] Shahar Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *43rd ACM Symposium on Theory of Computing (STOC)*, pages 139–148, 2011.

[4] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1064–1073, 2006.


[3] Under the stronger requirement that valuations support demand queries, things are different. Recall that demand queries can be used, for example, to solve the LP relaxation from Lecture #6. There is a polynomial-time DSIC mechanism that uses demand queries and obtains a $O(\log m \log \log m)$-approximation of the optimal welfare [2]; this is impossible with value queries only. It remains open whether or not there is a polynomial-time DSIC mechanism with demand queries that achieves a constant-factor approximation.

Without incentive constraints, there is a polynomial-time algorithm that uses demand queries and has approximation factor slightly better than $1 - \frac{1}{e}$ [7]; no such algorithm can obtain an approximation factor better than $1 - \frac{1}{2e}$, however [1].

[5] S. Dughmi, T. Roughgarden, and Q. Yan. From convex optimization to randomized mechanisms: toward optimal combinatorial auctions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 149–158, 2011.

[6] Shaddin Dughmi and Jan Vondrák. Limitations of randomized mechanisms for combinatorial auctions. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 502–511, 2011.

[7] Uriel Feige and Jan Vondrák. The submodular welfare problem with demand queries. *Theory of Computing*, 6(1):247–290, 2010.

[8] Subhash Khot, Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.

[9] J. Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–74, 2008.

# A Solving the Convex Program

We asserted in the main lecture that convex programs can be solved in polynomial time. This is only mostly true, and is subject to the following caveats.

1. One needs some handle on how big an optimal solution might be. This is not an issue for the convex program in this lecture since the $y_{ij}$'s are all bounded between 0 and 1.

2. The objective function should be computable in polynomial time. For the objective function (8), this is straightforward to do in time polynomial in the number of bits of precision needed.[4]

3. The gradient of the objective function should be computable in polynomial time. Because (8) is essentially the exponential function, it is its own gradient, and can again by evaluated in time polynomial in the number of bits of precision needed.

4. Most annoyingly, convex programs generally have only irrational optima and hence cannot be solved exactly in polynomial time. Under the above three assumptions, for any $L \geq 1$, the first $L$ bits of an optimal solution can be computed in time polynomial in $L$ and the problem size.

[4]We mentioned earlier that the results of this lecture can be extended from coverage valuations to valuations that are convex combinations of gross substitutes valuations (CGS). For abstract CGS valuations, computing the objective function (2) boils down to a randomized version of a value oracle: given a probability $x_j$ per item $j$, what is the expected valuation of $i$ for a random bundle, where each item $j$ lies in the bundle independently with probability $x_j$?

The three conditions above hold in the optimization problem (2)–(3); it remains to deal with the fourth issue. The first approach is to suffer a small error in the DSIC guarantee, analogous to the discretization error we incurred in our ascending auctions in Part I of the course. The second approach is to compute the solution to (2)–(3) only to the precision required to actually implement the rounding algorithm $R$. This idea leads to an exactly DSIC mechanism, at the expense of polynomial expected running time.