CS 369P: Polyhedral techniques in combinatorial optimization
Instructor: Jan Vondrák
Lecture date: November 30, 2010

1 The Submodular Welfare Problem

Given: m items, n agents with monotone submodular valuation functions $w_i : 2^{[m]} \to \mathbb{R}_+$.

Goal: Allocate items to agents in order to maximize $\sum_{i=1}^n w_i(S_i)$, where S_i is the set allocated to agent i.

Observe that this includes as a special case the budget-additive allocation problem, as well the allocation problem where the valuation functions are of coverage type etc. As we saw, this problem can be also formulated as a submodular maximization problem

$$\max\{f(S) : S \in \mathcal{I}\}$$

where $([n] \times [m], \mathcal{I})$ is a partition matroid defined by

$$\mathcal{I} = \{S \subseteq [n] \times [m] : \forall j; |S \cap \{(i,j) : 1 \leq j \leq m\}| \leq 1\}$$

and

$$f(S) = \sum_{i=1}^n w_i(\{j : (i,j) \in S\}).$$

1.1 Multilinear relaxation

We consider the following relaxation of the problem:

$$\max\{F(\mathbf{x}) : \mathbf{x} \in P(\mathcal{M})\}$$

where $F(\mathbf{x}) = \mathbb{E}[f(\hat{\mathbf{x}})]$ is the multilinear relaxation of f, and $P(\mathcal{M})$ is the matroid polytope corresponding to $\mathcal{M} = ([n] \times [m], \mathcal{I})$. Let's see more concretely what F and P look like. The ground set is $X = [n] \times [m]$, i.e. variables are naturally indexed by pairs (i,j) where i is an agent and j is an item. We have

$$F(\mathbf{x}) = \mathbb{E}[f(\hat{\mathbf{x}})] = \sum_{i=1}^n \mathbb{E}[w_i(R_i)]$$

where R_i contains each item j independently with probability x_{ij}. The matroid polytope $P(\mathcal{M})$ has the following form here:

$$P(\mathcal{M}) = \{\mathbf{x} \geq 0 : \forall j; \sum_{i=1}^n x_{ij} \leq 1\}.$$

Therefore, any point in P can be interpreted as a probability distribution: we allocate item j to agent i with probability x_{ij} (and possibly do not allocate it at all with some probability). The expected value of this random allocation is exactly $F(\mathbf{x})$, therefore if we want to achieve a certain approximation in expectation, it is sufficient to solve the continuous problem $\max\{F(\mathbf{x}) : \mathbf{x} \in P\}$.

1.2 The continuous greedy algorithm

It remains to show how to solve (approximately) the continuous problem $\max\{F(\mathbf{x}) : \mathbf{x} \in P\}$, where F is the multilinear relaxation of a monotone submodular function. We will not assume much about P, except that it is a *solvable polytope*.

Definition 1 *A class of polytopes is called* solvable *if there is an algorithm which for any polytope $P \subset \mathbb{R}^n$ in the class and any $\mathbf{w} \in \mathbb{R}^n$ solves the problem $\max\{\mathbf{w} \cdot \mathbf{x} : \mathbf{x} \in P\}$ in time polynomial in n and the bit-size of $\mathbf{w}$.*

In other words, we assume that we know how to optimize linear functions over P. (This is certainly true for our matroid polytope $P(\mathcal{M})$.) However, F is not linear, or even concave, so standard optimization techniques do not apply here. We describe a *continuous greedy algorithm* which is designed specifically for the problem $\max\{F(\mathbf{x}) : \mathbf{x} \in P\}$. The continuous greedy algorithm attacks this problem in a way similar to the discrete greedy algorithm: in each step, it tries to make a small improvement of maximum possible benefit. However, in contrast, it achieves a $(1 - 1/e)$-approximation for *any solvable polytope* P, i.e. in a much more general setting than a cardinality constraint. In a sense this is analogous to the fact that linear programming can be solved optimally for any (solvable) polytope, while discrete optimization problems can be solved optimally only for very special combinatorial structures.

In a compact form, the algorithm is described as follows.

Algorithm **ContinuousGreedy**(F, P):
Initialize $\mathbf{x}(0) := \mathbf{0} \in \mathbb{R}^N$;
Define $\mathbf{v}_{max}(\mathbf{x}) = \mathrm{argmax}_{\mathbf{v} \in P}(\mathbf{v} \cdot \nabla F(\mathbf{x}))$;
For $t \in [0, 1]$ solve
$\frac{d\mathbf{x}}{dt} = \mathbf{v}_{max}(\mathbf{x}(t))$;
Return $\mathbf{x}(1)$.

In words, the continuous greedy algorithm at time t moves in a direction $\mathbf{v}_{max}(\mathbf{x}(t))$ which maximizes the local gain $\mathbf{v} \cdot \nabla F(\mathbf{x})$ over all directions $\mathbf{v} \in P$. Importantly, we are able to find $\mathbf{v}_{max}(\mathbf{x}) = \mathrm{argmax}_{\mathbf{v} \in P}(\mathbf{v} \cdot \nabla F(\mathbf{x}(t))$ efficiently in each step, since this is just a linear optimization problem over P. The actual algorithm will proceed in finite steps, increasing t by some fixed small increment $\delta > 0$. However, we view the algorithm as truly continuous, which makes the analysis easier.

We remark that it is instructive to assume that P is a *down-monotone polytope*, i.e. $0 \leq \mathbf{x} \leq \mathbf{y} \in P \Rightarrow \mathbf{x} \in P$. This is without loss of generality, as F is monotone and hence P can be replaced by its down-monotone closure $P^{\downarrow} = \{\mathbf{y} \geq 0 : \exists \mathbf{x} \in P, \mathbf{y} \leq \mathbf{x}\}$ without affecting the optimum. Then, the initial point $\mathbf{0}$ is in $P^{\downarrow}$ and the trajectory $\mathbf{x}(t)$ is contained in $P^{\downarrow}$ as well. Nevertheless, the assumption of down-monotonicity is not used anywhere in the analysis. First, let us prove the following lemma.

Lemma 2 *For any monotone submodular function $f : 2^N \to \mathbb{R}_+$, its multilinear extension $F : [0, 1]^N \to \mathbb{R}_+$, and a polytope $P \subseteq [0, 1]^N$, let $\mathbf{x} \in [0, 1]^N$ be an arbitrary point and $OPT = \max\{F(\mathbf{y}) : \mathbf{y} \in P\}$. Then there is $\mathbf{v} \in P$ such that*

$$\mathbf{v} \cdot \nabla F(\mathbf{x}) \geq OPT - F(\mathbf{x}).$$

Proof: We claim that any $\mathbf{v} \in P$ such that $OPT = F(\mathbf{v})$ satisfies the conclusion of the lemma. Consider a line from $\mathbf{x}$ towards $\mathbf{v} \vee \mathbf{x}$; this line has direction $\mathbf{d} = (\mathbf{v} \vee \mathbf{x}) - \mathbf{x} = (\mathbf{v} - \mathbf{x}) \vee 0$, which satisfies $0 \leq \mathbf{d} \leq \mathbf{v}$. Let $\phi(\xi) = F(\mathbf{x} + \xi\mathbf{d})$ denote the objective function along this line. The starting point has value $\phi(0) = F(\mathbf{x})$. Now consider the point corresponding to $\xi = 1$; its value is $\phi(1) = F(\mathbf{x} + \mathbf{d}) \geq OPT$, because $\mathbf{x} + \mathbf{d} = \mathbf{v} \vee \mathbf{x} \geq \mathbf{v}$ and F is non-decreasing in each coordinate. As we saw in the last lecture, $\phi(\xi)$ is concave. This means that

$$\phi(1) - \phi(0) \leq \phi'(0) = \mathbf{d} \cdot \nabla F(\mathbf{x}).$$

Using $\mathbf{d} \leq \mathbf{v}$ and the nonnegativity of ∇F (due to monotonicity), we obtain

$$\mathbf{v} \cdot \nabla F(\mathbf{x}) \geq \mathbf{d} \cdot \nabla F(\mathbf{x}) \geq \phi(1) - \phi(0) \geq OPT - F(\mathbf{x}).$$

□

Recall that $\mathbf{v} \cdot \nabla F(\mathbf{x})$ is the derivative of F along a line of direction $\mathbf{v}$ at the point $\mathbf{x}$. The continuous greedy algorithm chooses a direction maximizing this quantity over $\mathbf{v} \in P$. I.e., what we proved is that at each time t, the local gain in terms of F is at least $OPT - F(\mathbf{x}(t))$. This leads to a differential equation whose solution yields the factor of $1 - 1/e$.

Theorem 3 *For any monotone submodular function $f : 2^N \to \mathbb{R}_+$, its multilinear extension $F : [0,1]^N \to \mathbb{R}_+$, and any solvable polytope $P \subseteq [0,1]^N$, the continuous greedy algorithm finds a point $\mathbf{x}(1) \in P$ such that*

$$F(\mathbf{x}(1)) \geq (1 - 1/e)OPT$$

where $OPT = \max\{F(\mathbf{x}) : \mathbf{x} \in P\}$.

Proof: By design of the continuous greedy algorithm, we obtain

$$\mathbf{x}(1) = \int_0^1 \mathbf{v}_{max}(\mathbf{x}(t))dt.$$

Therefore, $\mathbf{x}(1)$ is a convex linear combination of vectors $\mathbf{v}_{max}(\mathbf{x}(t)) \in P$, and hence $\mathbf{x}(1) \in P$.

To prove the approximation guarantee, let us analyze $F(\mathbf{x}(t))$. By the chain rule and the design of the continuous greedy algorithm,

$$\frac{d}{dt}F(\mathbf{x}(t)) = \frac{d\mathbf{x}}{dt} \cdot \nabla F(\mathbf{x}(t)) = \mathbf{v}_{max}(\mathbf{x}(t)) \cdot \nabla F(\mathbf{x}(t)) = \max_{\mathbf{v} \in P} \mathbf{v} \cdot \nabla F(\mathbf{x}(t)).$$

By Lemma 2, we know that this quantity is at least $OPT - F(\mathbf{x}(t))$. Hence, we obtain

$$\frac{d}{dt}F(\mathbf{x}(t)) + F(\mathbf{x}(t)) \geq OPT.$$

This differential inequality can be solved as follows. Multiplying by e^t, we obtain:

$$\frac{d}{dt}(e^t F(\mathbf{x}(t))) = e^t \frac{d}{dt}F(\mathbf{x}(t)) + e^t F(\mathbf{x}(t)) \geq e^t OPT.$$

The initial condition is $F(\mathbf{x}(0)) = 0$ which implies

$$e^t F(\mathbf{x}(t)) \geq \int_0^t e^\tau \, OPT \, d\tau = (e^t - 1)OPT.$$

Therefore, the solution at time t satisfies $F(\mathbf{x}(t)) \geq (1 - e^{-t})OPT$ and at time $t = 1$ we get $F(\mathbf{x}(1)) \geq (1 - 1/e)OPT$. □

As we noted, an actual implementation needs to emulate the continuous greedy algorithm in a finite number of steps, in some sense numerically solving the differential equation $\frac{d\mathbf{x}}{dt} = \mathbf{v}_{max}(\mathbf{x}(t))$. This can be done by standard techniques, for example by choosing a fixed time increment $\delta > 0$ and approximating the process by $\mathbf{x}(t+\delta) = \mathbf{x}(t) + \delta \mathbf{v}_{max}(\mathbf{x}(t))$. It can be shown that this deviates from the analysis above by an error which tends to zero as $\delta \to 0$.