Monotone Submodular Maximization over a Matroid

Yuval Filmus*

October 23, 2014

1 Introduction: Maximum Coverage

One of the classical NP-hard problems is Maximum Coverage, the problem dual to Set Cover. In this problem we are given the following pieces of data:

- A universe U and an associated non-negative weight function $w\colon U \to \mathbb{R}_{\geq 0}$.
- A collection $\mathcal{F}$ of subsets of U.
- A number r.
- Goal: choose r sets from $\mathcal{F}$ which together cover elements of U with maximal total weight (as measured according to w).

We extend the definition of w to subsets of U linearly, and to subsets of $\mathcal{F}$ by defining $w(S) = w(\bigcup S)$ for $S \subseteq \mathcal{F}$.

Already from Karp's time it had been known that this problem is NP-hard, for the same reason that Set Cover is. The classical greedy algorithm provides a $1 - 1/e$ approximation for Maximum Coverage:

- Let $S^0 \leftarrow \emptyset$.
- For $i = 1, \ldots, r$, let $S^i \leftarrow S^{i-1} + x_i$, where x_i maximizes $w(S^i)$.
- Return $S = S^r$.

It is a classical result that $w(S) \geq (1 - 1/e)w(O)$ for *any* collection O of r subsets from $\mathcal{F}$, and so this algorithm is a $1 - 1/e$ approximation. Feige showed in 1998 that this is best-possible: for any $\epsilon > 0$, if Maximum Coverage can be approximated within $1 - 1/e + \epsilon$ in polynomial time then $P = NP$.

1.1 Monotone Submodular functions

The goal of Maximum Coverage is to choose a subset S of $\mathcal{F}$ of size r maximizing $w(S)$. When analyzing the greedy algorithm, we use the fact that w is a *coverage function*, that is it measures the total weight of elements covered by the sets in its input. However, it turns out that a different analysis works given only the following properties, all satisfied by coverage functions:

- Normalization: $w(\emptyset) = 0$. (In fact, it is enough to assume $w(\emptyset) \geq 0$.)
- Monotonicity: if $A \subseteq B \subseteq \mathcal{F}$ then $w(A) \leq w(B)$.
- Submodularity: if $A \subseteq B \subseteq \mathcal{F}$ and $x \in \mathcal{F}$ then $w(A + x) - w(A) \geq w(B + x) - w(B)$.

*Joint work with Justin Ward.

We can think of submodularity as saying that the second discrete derivative is non-positive. Coverage functions satisfy similar conditions for discrete derivatives of any order, but the properties of the first two derivatives suffice to analyze the greedy algorithm. In other words, the greedy algorithm is provides a $1-1/e$ approximation to the following problem, Max Monotone Submodular:

- An abstract universe $\mathcal{F}$.
- An oracle to a normalized, monotone submodular set function f on $\mathcal{F}$.
- A number r.
- Goal: choose a subset S of $\mathcal{F}$ of size r maximizing $f(S)$.

Nemhauser, Fisher and Wolsey showed that any algorithm providing a $1 - 1/e + \epsilon$ approximation for Max Monotone Submodular must query the oracle exponentially (in r) many times; this is known as a value oracle lower bound.

2 The Plot Thickens: Max SAT

Another well-known NP-hard optimization is Max SAT, which is one possible optimization version of SAT:

- A set U of clauses with associated non-negative weights $w\colon U \to \mathbb{R}_{\geq 0}$.
- Goal: find a truth assignment maximizing the weight of satisfied clauses.

We can think of Max SAT as a constrained version of Maximum Coverage. The idea is as follows. For each literal $x_i^{\pm}$, let $C_i^{\pm}$ be the set of clauses satisfied by the assignment x, and let $\mathcal{F}$ be the collection of all C_x. If there are r variables $x_1, \ldots, x_r$, then each assignment corresponds to a subset of $\mathcal{F}$ of size r in which out of C_i^+, C_i^- exactly one set is chosen. The goal is to find such a subset S maximizing $w(S)$. More generally, consider the following problem, Partition Maximum Coverage:

- A universe U and an associated non-negative weight function $w\colon U \to \mathbb{R}_{\geq 0}$.
- An integer r and subsets $\mathcal{F}_1, \ldots, \mathcal{F}_r$ of U.
- Goal: choose one set out of each $\mathcal{F}_i$ maximizing the total weight of elements covered.

How well does the greedy algorithm work in this setting? Consider the following instance:

$$\underbrace{x_1 \vee \bar{x}_2}_{1} \vee \underbrace{\bar{x}_1}_{1} \vee \underbrace{x_1}_{\epsilon} \vee \underbrace{x_2}_{\epsilon}.$$

The greedy algorithm starts by setting $x_1 = \top$, satisfying clauses of total weight $1 + \epsilon$. It then sets $x_2 = \top$, resulting in a solution of value $1 + 2\epsilon$. However, the optimal assignment $x_1 = x_2 = \bot$ has value 2. So the approximation ratio of the greedy algorithm is at most $1/2$. It turns out that the approximation ratio is also always at least $1/2$.

Another approach which doesn't work is *local search*:

- Start with some feasible solution S, say the greedy solution.
- Repeat: Find some $A \subseteq S$ and $B \cap S = \emptyset$ of size $|A| = |B| \leq k$ such that $w(S \setminus A \cup B) > w(S)$, and put $S \to S \setminus A \cup B$.
- Eventually no such improvement (A, B) exists. Return S.

Here k is a parameter of the algorithm. If $|\mathcal{F}| = n$, then the running time of each step is $O(n^k)$, so for this to be polynomial time, k needs to be constant. Considering our former example, for $k = 1$ the assignment $x_1 = x_2 = \top$ is locally optimal, that is no exchange (A, B) with $|A| = |B| \leq 1$ improves the weight of satisfied clauses. More generally, it can be shown that the approximation ratio of this algorithm is $(r-1)/(2r-k-1) = 1/2 + O(k/r)$, so for constant k this does not result in a constant improvement.

2.1 Continuous Greedy algorithm

Vondrák et al. came up with an algorithm which gives the optimal approximation ratio $1 - 1/e$ for Partition Maximum Coverage. In fact, their algorithm works in a more general setting. First of all, again we can replace the coverage function w with an arbitrary normalized monotone submodular function f given by an oracle. Second, the partition constraint can be replaced by a more general type of constraint known as a matroid constraint. A matroid $\mathfrak{m}$ on a set $\mathcal{F}$ is a non-empty collection of subsets of $\mathcal{F}$ satisfying the following two properties:

- Hereditarity: If $B \in \mathfrak{m}$ and $A \subseteq B$ then $A \in \mathfrak{m}$.
- Exchange property: if $A, B \in \mathfrak{m}$ and $|A| < |B|$ then $A + x \in \mathfrak{m}$ for some $x \in B \setminus A$.

The sets in $\mathfrak{m}$ are known as *independent sets*, and maximal independent sets are known as *bases*. All bases in a matroid have the same size called the *rank*. For example, in a partition matroid corresponding to the partition $\mathcal{F} = \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_r$ a set is independent if it contains at most one set out of each $\mathcal{F}_i$, and a base contains exactly one set out of each $\mathcal{F}_i$; the rank is r. The general problem solved by Vondrák et al. is Matroid Max Monotone Submodular:

- An abstract universe $\mathcal{F}$.
- An oracle to a normalized, monotone submodular set function f on $\mathcal{F}$.
- An oracle to a matroid $\mathfrak{m}$ on $\mathcal{F}$.
- Goal: choose an independent set $S \in \mathfrak{m}$ maximizing $f(S)$. (Due to monotonicity, we can assume that S is in fact a base.)

One general idea behind the continuous greedy algorithm is to consider a continuous relaxation of the original problem. Starting with the function f on $\mathcal{F}$, we define its multilinear relaxation F on $[0,1]^{\mathcal{F}}$ as follows: $F(X) = f(x)$, where x is sampled by putting $\alpha \in \mathcal{F}$ inside with probability X_α. Written explicitly, the function F is multilinear in its inputs. The properties of monotonicity and submodularity translate to F being monotone and concave (having non-positive second derivatives). Next, consider the matroid polytope $P \subseteq [0,1]^{\mathcal{F}}$, which is the convex hull of the characteristic vectors of all independent sets in $\mathfrak{m}$. In the special case of a partition matroid, P is given by the constraints $\sum_{\alpha \in \mathcal{F}_i} X_\alpha \leq 1$ for all i, in addition to non-negativity $X_\alpha \geq 0$. The continuous greedy algorithm is composed of two steps:

- Continuous greedy (proper): find $X \in P$ satisfying $F(X) \geq (1 - 1/e)F(O)$, where O is the optimal solution vector (in P).
- Rounding: find a subset $x \in \mathfrak{m}$ such that $f(x) \geq F(X)$.

In the case of a partition matroid, the rounding step is very easy: if we choose x according to X (like in the definition of F), then $\mathbb{E}\, f(x) \geq \mathbb{E}\, F(X)$. The general case is more complicated, and requires the use of either pipage rounding or the more efficient swap rounding, which we don't describe. As for the first stage, it is a discretized version of the following algorithm running in continuous time:

- Start with $X(0) = 0 \in P$.
- At time $t \in [0,1]$, let $X(t)' = 1_B$ for the base B maximizing $F(X(t))'$.
- Return $X(1)$.

The optimization in the second step can be done since $F(X(t))'$ is linear in $X(t)'$, and linear functionals can be maximized exactly using the greedy algorithm. The idea behind the analysis of the algorithm is showing that $F(X(t))' \geq f(O) - F(X(t))$, by considering the choice O for B. This shows that $F(X(t))$ dominates the solution to the differential equation $y' \geq f(O) - y$ with initial value $y(0) = 0$, whose solution is $y(t) = (1 - e^t)f(O)$. Putting $t = 1$, we obtain $y(1) = (1 - 1/e)f(O)$.

In practice, the continuous-time algorithm is discretized in order to obtain a polynomial-time algorithm. One problem, however, is that the function F cannot be evaluated exactly in polynomial time, unless f has some known structure (for example, f is a coverage function). Instead, F has to be approximated by sampling. As a result, the algorithm is randomized.

A recent fast variant of the continuous greedy algorithm, due to Vondrák and Ashwinkumar Badanidiyuru runs in time essentially $O(n^2 \log^2 n)$.

There are a few reasons to be unhappy with the continuous greedy algorithm:

1. The original implementation of the algorithm was complicated, especially the rounding procedure for general matroids (pipage rounding).
2. The original implementation of the algorithm was slow.
3. The algorithm is not combinatorial: its intermediate objects are fractional.
4. The algorithm is randomized (randomization is needed to estimate F).

As we mentioned, Vondrák and Ashwinkumar fixed the first two problems. Regarding the second problem, Filmus and Feldman have independently come up with a combinatorial version which applies swap rounding at each step to reduce the fractional solution to a multiple of a single basis. Even the remaining goal, derandomizing the algorithm, has not been achieved, but the resulting algorithm is still interesting.

3 Our algorithm: Maximum Coverage

The greedy algorithm for Maximum Coverage is simple and combinatorial. In contrast, the continuous greedy algorithm uses a continuous relaxation, and as a result it is randomized rather than deterministic. Is there a different, combinatorial algorithm that solves the problem instead? This is the motivation behind our algorithm. Our main idea is to use the paradigm of non-oblivious local search, due to Alimonti and to Khanna, Motwani, Sudan and U. Vazirani. In this paradigm, we use a local search algorithm, pretending to optimize some auxiliary objective function g instead of the actual function f:

- Start with some feasible solution S, say the greedy solution.
- Repeat: Find some $a \in S$ and $b \notin S$ such that $g(S - A + B) > g(S)$, and put $S \to S - A + B$.
- Eventually no such improvement (A, B) exists. Return S.

The hope is that for a proper choice of g, the resulting solution is a good approximate solution *in terms of* f.

Consider again our bad example:

$$\underbrace{x_1 \vee \bar{x}_2}_{1} \vee \underbrace{\bar{x}_1}_{1} \vee \underbrace{x_1}_{\epsilon} \vee \underbrace{x_2}_{\epsilon}.$$

The greedy solution is $x_1 = x_2 = \top$. If we set $x_1 = \bot$ then we lose the third clause, while if we set $x_2 = \bot$ then we lose the fourth clause; however, in that case we do gain something else: the first clause is now doubly satisfied. This is advantageous in terms of local search, since whatever happens in the following step, this clause will remain satisfied. This suggests giving more weight to clauses satisfied more than once. More generally, we consider the auxiliary objective function

$$g(S) = \sum_{x \in \bigcup S} w(x)\alpha_{N(x,S)},$$

where $N(x, S)$ is the number of times that x appears in sets in S, i.e. the number of times x is covered. How do we choose the weights α_i, and what is the resulting approximation ratio? Suppose that the algorithm produces the output $S = \{S_1, \ldots, S_n\}$, where S_i belongs to the ith partition, and let $O = \{O_1, \ldots, O_n\}$ be an optimal solution. The approximation ratio is given by the following linear program:

$$\begin{aligned} &\min f(S) \\ s.t. &f(O) = 1 \\ &f(S) \geq f(S - S_i + O_i) \text{ for } i = 1, \ldots, r \end{aligned}$$

The variables in this program are the weights of all areas in the Venn diagram of the underlying set cover instance. In fact, a symmetrization argument shows that the following linear program has the same solution:

$$\begin{aligned} &\min f(S) \\ s.t. &f(O) = 1 \\ &\sum_{i=1}^{r} [f(S) - f(S - S_i + O_i)] \geq 0 \end{aligned}$$

This allows us to reduce the number of variables: now it is enough to know, for each ℓ, c, g, the total weight in areas belonging to exactly sets of the following form: $S_1, \ldots, S_\ell, O_{\ell+1}, \ldots, O_{\ell+g}, S_{\ell+g+1}, O_{\ell+g+1}, \ldots, S_{\ell+g+c}, O_{\ell+g+c}$.

In order to find the optimal weights $\alpha_1, \ldots, \alpha_r$, we need to maximize over $\alpha_1, \ldots, \alpha_r$. Dualizing the inner linear program and setting arbitrarily the value of one of the dual variables (which corresponds to a scaling of the α_i sequence), we obtain the following linear program:

$$\begin{aligned} &\max \theta \\ s.t. &1_{g+c \geq 1}\theta + ((\ell + g)\alpha_{\ell+c} - \ell\alpha_{\ell+c-1} - g\alpha_{\ell+c+1}) \leq 1_{\ell+c \geq 1} \text{ for } 1 \leq \ell + c + g \leq r \end{aligned}$$

Here $\alpha_0 = 0$. For each reasonably small value of r, we can solve this program on a computer, and find the optimal θ and $\alpha_1, \ldots, \alpha_r$. When doing that, we were astonished to find out that $\theta \to 1 - 1/e$! It is natural to consider the corresponding infinite linear program, in which the constraints for all ℓ, c, g are present. By considering the tight constraints for each r, it is not difficult to come up with the solution for the infinite linear program: $\theta = 1 - 1/e$ and

$$\alpha_0 = 0, \quad \alpha_1 = 1 - \frac{1}{e}, \quad , \alpha_{i+1} = (i+1)\alpha_i - i\alpha_{i-1} - \frac{1}{e}.$$

Asymptotically, $\alpha_i \approx (1/e) \log i$. Moreover, the sequence α_i is increasing, and its difference sequence is decreasing. These two facts allow us to show that the sequence is feasible for the infinite linear program. Iterated difference sequences are alternatingly increasing and decreasing, mirroring the properties of successive derivatives of the logarithm function.

3.1 More details

Let $w(\ell, c, g)$ be the total weight in areas belonging to exactly ℓ sets S_i, g sets O_i, and c sets S_i, O_i (all indices being disjoint). In terms of these variables, we can write the linear program for the approximation ratio as

$$\begin{aligned}
&\min \sum_{\ell+c\geq 1} w(\ell,c,g) \\
s.t. &\sum_{g+c\geq 1} w(\ell,c,g) \geq 1 \\
&\sum_{\ell,c,g} w(\ell,c,g)[\ell(\alpha_{\ell+c}-\alpha_{\ell+c-1})+g(\alpha_{\ell+c}-\alpha_{\ell+c+1})] \geq 0
\end{aligned}$$

For brevity, we define $\beta_\ell = \alpha_{\ell+1} - \alpha_\ell$. Calculting the dual, we associate a variable $\theta \geq 0$ with the first equation, and a variable $\zeta \geq 0$ with the second, obtaining

$$\begin{aligned}
&\max \theta \\
s.t. &\zeta(\ell\beta_{\ell+c-1} - g\beta_{\ell+c}) + \theta[g+c\geq 1] \leq [\ell+c\geq 1] \text{ for all } \ell,c,g \geq 0
\end{aligned}$$

We can choose $\zeta = 1$ arbitrarily, since this just amounts to some particular scaling of the weights. Combining this with the outer maximization, we obtain a linear program for the α_i (in the guise of the β_i; we choose $\alpha_0 = 0$ arbitrarily):

$$\begin{aligned}
&\max \theta \\
s.t. &\ell\beta_{\ell+c-1} - g\beta_{\ell+c} + \theta[g+c\geq 1] \leq [\ell+c\geq 1] \text{ for all } \ell,c,g \geq 0
\end{aligned}$$

For any particular rank r, it is enough to consider $\ell + c + g \leq r$, but we will solve the infinite linear program. The solution is $\theta = 1 - 1/e$. The tight constraints are those corresponding to $(\ell, 0, 1)$, which are $\beta_0 = 1 - 1/e$ for $\ell = 0$ and $\beta_\ell = \ell\beta_{\ell-1} - 1/e$ for $\ell > 0$. Easy induction gives a closed-form expression for β_ℓ:

$$\beta_\ell = e^{-1} \sum_{t\geq 1} \frac{\ell!}{(\ell+t)!}.$$

This shows in particular that $0 < \beta_\ell < e^{-1}/\ell$, since $\sum_{t\geq 1} 1/(\ell+1)^t = 1/\ell$. Also, it shows that β_ℓ is decreasing and so $\beta_\ell \leq 1 - 1/e$.

It remains to show that the program is feasible. Since $\beta_\ell > 0$, it is enough to consider the cases $g = 0$ and $g = 1$. When $g = c = 0$ (and so $\ell > 0$) we have $\ell\beta_{\ell-1} = \beta_\ell + e^{-1} < 1$. When $c \geq 1$ we have $\ell\beta_{\ell+c-1} + 1 - e^{-1} < e^{-1}\ell/\ell + 1 - e^{-1} = 1$. When $g = 1$ it is enough to consider $c \geq 1$ (since otherwise the constraint is tight). Since $\beta_{\ell+c} = (\ell+c)\beta_{\ell+c-1} - e^{-1}$, we get $\ell\beta_{\ell+c-1} - \beta_{\ell+c} + \theta = -c\beta_{\ell+c-1} + 1 < 1$.

3.2 Polynomial-time algorithm?

The algorithm as stated is not quite polynomial time, since it might take many swaps to reach a local optimum. In fact, there is an explicit example involving a counter in which there is a sequence of exponentially many local improvements leading to a local optimum, for oblivious local search. (It might be, however, that if you start with the greedy solution and always choose the best improvement, then the algorithm always converges quickly.) In order to solve this potential problem, we modify our algorithm to perform an exchange only if it results in a relative $(1+\epsilon)$-improvement, for some appropriately chosen, polynomially small ϵ. The algorithm thus terminates at an approximate local maximum, and we have to generalize slightly the former analysis. The end result is that the algorithm is a $1 - 1/e - O(\epsilon/n \log n)$ approximation. By choosing ϵ small enough, we can obtain a $1 - 1/e - O(1/n)$ approximation, which can be boosted to a clean $1 - 1/e$ approximation by "guessing" a constant number of sets from the optimal solution.

3.3 General matroids

So far we have described and analyzed the algorithm only for partition matroids. However, it turns out that partition matroids are the general case as far as our algorithm is concerned. Indeed, a simple result of Brualdi shows that given any two bases S, O of a matroid, it is always possible to find a bijection $\pi\colon S \to O$ such that for all $x \in S$, $S - x + \pi(x)$ is a base. This is exactly the condition we needed in the analysis of our algorithm.

4 Our algorithm: general case

We can repeat the same analysis we did in the special case for this general case. Instead, consider some monotone submodular function f given as a value oracle, but secretly being a coverage function. Using inclusion-exclusion, we can calculate $w(A_1 \cap \cdots A_k \cap \overline{B_1} \cap \cdots \cap \overline{B_l})$ given the value of f on subsets of $\{A_1, \ldots, A_k, B_1, \ldots, B_l\}$, and through that, we get an expression for $g(S)$ which mirrors the previous one:

$$g(S) = \sum_{T \subseteq S} \sum_{i=r-|T|}^{r} (-1)^{r+i+|T|+1} \frac{|T|!}{(r-i)!(i+|T|-r)!} \alpha_i f(T).$$

If we choose the same α_i sequence as before divided by $1 - 1/e$, then amazingly enough we get the following alternative expression for g:

$$g(S) = \sum_{\emptyset \neq T \subseteq S} \int_0^1 p^{|T|-1}(1-p)^{|S|-|T|} \frac{e^p}{e-1}\, dp f(T).$$

The expression becomes nicer if we consider instead the *marginals* $f_A(x) = f(A+x) - f(A)$ and $g_A(x) = g(A+x) - g(A)$, and further assume that $f(\emptyset) = 0$:

$$\begin{aligned} g_S(x) &= \sum_{T \subseteq S} \int_0^1 \left(p^{|T|-1}[(1-p)^{|S|+1-|T|} - (1-p)^{|S|-|T|}]f(T) + p^{|T|}(1-p)^{|S|-|T|} f(T+x)\right) \frac{e^p}{e-1}\, dp \\ &= \sum_{T \subseteq S} \int_0^1 p^{|T|}(1-p)^{|S|-|T|} \frac{e^p}{e-1}\, dp f_T(x). \end{aligned}$$

In other words, if we let $p \sim P$, where P is the distribution supported on $[0,1]$ with density $e^p/(e-1)$, and let $T \subseteq S$ be chosen by putting in each element with probability p (a distribution we denote by $B_p(S)$), then $g_S(x) = \mathbb{E}\, f_T(x)$. The corresponding distribution of T we denote by μ_S. We come back later to this particular form of g.

As in the case of the greedy algorithm, although now we have obtained the "same" algorithm, its proof of correctness in the general case is different from the proof of correctness in the case of Maximum Coverage. We start by establishing some simple properties of the function g. Since f is monotone, $g_S(x) = \mathbb{E}\, f_T(x) \geq 0$, showing that g is also monotone. Submodularity of f shows that if $S_1 \subseteq S_2$ then

$$g_{S_2}(x) = \mathop{\mathbb{E}}_{p \sim P} \mathop{\mathbb{E}}_{T \sim B_p(S_1)} \mathop{\mathbb{E}}_{U \sim B_p(S_2 \setminus S_1)} f_{T \cup U}(x) \leq \mathop{\mathbb{E}}_{p \sim P} \mathop{\mathbb{E}}_{T \sim B_p(S_1)} f_T(x) = g_{S_1}(x),$$

showing that g is also submodular. In the same way, one can show that higher-order properties of f carry over to g. The analysis of the algorithm then rests on the following fundamental inequality:

$$\frac{e}{e-1} f(S) \geq f(O) + \sum_{i=1}^{r} [g(S) - g(S - S_i + O_i)]. \tag{1}$$

The idea of the proof is to consider the quantity $X = \sum_{i=1}^r g_{S-S_i}(S_i)$. We start by showing that

$$\sum_{i=1}^r g_{S-S_i}(S_i) \geq \sum_{i=1}^r [g(S) - g(S - S_i + O_i)] + \mathop{\mathbb{E}}_{T\sim\mu_S} \sum_{i=1}^r f_{T-O_i}(O_i)$$
$$\geq \sum_{i=1}^r [g(S) - g(S - S_i + O_i)] + f(O) - \mathop{\mathbb{E}}_{T\sim\mu_S} f(T).$$

On the other hand,

$$\sum_{i=1}^r g_{S-S_i}(S_i) = \frac{e}{e-1} f(S) - \mathop{\mathbb{E}}_{T\sim\mu_S} f(T).$$

Putting these things together completes the proof.

The proofs of the first two inequalities use the monotonicity and submodularity of g. The proof of the formula for X, in turn, requires properties of the distributions μ. Specifically, the coefficient of $f(T)$ in X is exactly $|T|\mu_{r-1}(|T|-1) - (r-|T|)\mu_{r-1}(|T|)$, where we replaced $\mu_A(B)$ with $\mu_{|A|}(|B|)$ since the probabilities indeed depend only on these cardinalities. This expression satisfies the following formula:

$$\mu_a(b) = (a-b)\mu_{a-1}(b) - b\mu_{a-1}(b-1) + \begin{cases} -1/(e-1) & \text{if } b = 0, \\ 0 & \text{if } 0 < b < a, \\ e/(e-1) & \text{if } b = a. \end{cases} \tag{2}$$

Where does the form of g come from? When proving that g is monotone, we made use of the fact that $g_S(x) = \sum_{T\subseteq S} \mu_S(T) f_T(x)$ for some $\mu_S(T) \geq 0$. When proving that g is submodular, we used the fact that $\mu_{S_1}(T) = \sum_{U\subseteq S_2\setminus S_1} \mu_{S_2}(T \cup U)$. In particular, this implies that $\sum_{T\subseteq S} \mu_S(T)$ is constant. Choosing this arbitrary constant to be 1, we obtain a probability distribution μ_S. When $S_1 \subseteq S_2$ then we have $\mu_{S_2} \cap S_1 = \mu_{S_1}$, and so the distribution of μ_{S_1} can be obtained from that of μ_{S_2}. In this way we can define a probability distribution μ_U on some countable set U. This probability distribution is exchangeable, and so de Finetti's theorem shows that μ_U is obtained by choosing $p \sim P$ (for some distribution P on $[0,1]$) and then $T \sim B_p(U)$, which explains the form of g. The distribution P is determined by formula (2).

As in the case of the continuous greedy algorithm, the function g cannot be evaluated exactly and has to be sampled. For this reason, our algorithm is randomized.

4.1 More details on the proof

For the proof we will need the following two easy results. The first result concerns the definition of submodularity, which can be stated as $f_A(x) \geq f_B(x)$ whenever $A \subseteq B$. We claim that we can replace x by a set X. Indeed, if $X = \{x_1, \ldots, x_\ell\}$ then

$$f_A(X) = \sum_{i=1}^\ell f_{A\cup\{x_1,\ldots,x_{i-1}\}}(x_i) \geq \sum_{i=1}^\ell f_{B\cup\{x_1,\ldots,x_{i-1}\}}(x_i) = f_B(X).$$

The second result states that the marginal value is sublinear: $f_A(X) + f_A(Y) \geq f_A(X \cup Y)$. Indeed,

$$f_A(X) + f_A(Y) \geq f_{A\cup Y}(X) + f_A(Y) = f_A(X \cup Y).$$

Here are some more details regarding the proofs of (1) and (2). We start with the proof of (1). The proof is divided into three parts:

1. For all i: $g_{S-S_i}(S_i) \geq g(S) - g(S - S_i + O_i) + \mathbb{E}_{T\sim\mu_S} f_{T-O_i}(O_i)$.
2. For all $T \subseteq S$: $\sum_{i=1}^r f_{T-O_i}(O_i) \geq f(O) - f(T)$.
3. $\sum_{i=1}^r g_{S-S_i}(S_i) = e/(e-1) f(S) - 1/(e-1) f(\emptyset) - \mathbb{E}_{T\sim\mu_S} f(T)$.

Combining all three parts, we get:

$$\begin{aligned}
&\frac{e}{e-1}f(S) - \frac{1}{e-1}f(\emptyset) - \mathop{\mathbb{E}}_{T\sim\mu_S} f(T) \\
&\overset{(3)}{=} \sum_{i=1}^{r} g_{S-S_i}(S_i) \\
&\overset{(1)}{\geq} \sum_{i=1}^{r} [g(S) - g(S - S_i + O_i)] + \mathop{\mathbb{E}}_{T\sim\mu_S} \sum_{i=1}^{r} f_{T-O_i}(O_i) \\
&\overset{(2)}{\geq} \sum_{i=1}^{r} [g(S) - g(S - S_i + O_i)] + f(O) - \mathop{\mathbb{E}}_{T\sim\mu_S} f(T).
\end{aligned}$$

We deduce (1) by cancelling $\mathbb{E}_{T\sim\mu_S} f(T)$ and using $f(\emptyset) \geq 0$.

For the proof of Lemma 1, we consider two cases: $S_i \notin O$ and $S_i \in O$ (in which case $S_i = O_i$). In the first case, we have

$$\begin{aligned}
g_{S-S_i}(S_i) &\geq g_{S-S_i+O_i}(S_i) \\
&= g(S + O_i) - g(S - S_i + O_i) \\
&= g_S(O_i) + g(S) - g(S - S_i + O_i) \\
&= g(S) - g(S - S_i + O_i) + \mathop{\mathbb{E}}_{T\sim\mu_S} f_T(O_i).
\end{aligned}$$

In the second case, we actually have equality:

$$\begin{aligned}
g_{S-S_i}(S_i) &= \mathop{\mathbb{E}}_{T\sim\mu_{S-S_i}} f_T(S_i) \\
&= \mathop{\mathbb{E}}_{T\sim\mu_S} f_{T-S_i}(S_i) \\
&= g(S) - g(S - S_i + O_i) + \mathop{\mathbb{E}}_{T\sim\mu_S} f_{T-S_i}(S_i).
\end{aligned}$$

Lemma 2 relies only on the submodularity of f:

$$\sum_{i=1}^{r} f_{T-O_i}(O_i) \geq \sum_{O_i\notin T} f_T(O_i) \geq f_T(O \setminus T) = f(O \cup T) - f(T) \geq f(O) - f(T).$$

Lemma 3 essentially follows from (2). Indeed,

$$\sum_{i=1}^{r} g_{S-S_i}(S_i) = \sum_{i=1}^{r} \sum_{T\subseteq S-S_i} \mu_{S-S_i}(T)[f(T + S_i) - f(T)].$$

The coefficient of $f(T)$ in this sum is

$$\sum_{S_i\in T} \mu_{S-S_i}(T - S_i) - \sum_{S_i\notin T} \mu_{S-S_i}(T) = |T|\mu_{r-1}(|T| - 1) - (r - |T|)\mu_{r-1}(|T|).$$

According to (2), this is equal to

$$\frac{[|T| = r]e - [|T| = 0]1}{e-1} - \mu_r(|T|),$$

which gives the lemma.

We now turn to the proof of (2). The proof is a simple application of integration by parts:

$$\begin{aligned}\mu_a(b) &= \int_0^1 p^b(1-p)^{a-b}\frac{e^p}{e-1}\,dp \\ &= p^b(1-p)^{a-b}\frac{e^p}{e-1}\Big|_0^1 - \int_0^1 (bp^{b-1}(1-p)^{a-b} - (a-b)p^b(1-p)^{a-b-1})\frac{e^p}{e-1}\,dp \\ &= \frac{[b=a]e - [b=0]1}{e-1} - b\mu_{a-1}(b-1) + (a-b)\mu_{a-1}(b).\end{aligned}$$

4.2 Relation to Maximum Coverage

We claimed that this algorithm reduces to the one for maximum coverage when f is a coverage function. To show this, notice first that if f is a coverage function then $g(S) = \sum_{x \in \bigcup S} w(x)\alpha_{N(x,S)}$ for some coefficients α_ℓ. In fact, to find these coefficients it will be simpler to consider an element x of unit weight and

$$g_S(\{x\}) = \alpha_{N(x,S)+1} - \alpha_{N(x,S)} = \beta_{N(x,S)}.$$

When $S = \emptyset$ we get $\beta_0 = g_\emptyset(\{x\}) = f_\emptyset(\{x\}) = 1$. Now consider a set S containing ℓ copies of $\{x\}$. Then $\beta_\ell = g_S(\{x\}) = \mathbb{E}_{T\sim\mu_S} f_T(\{x\}) = \Pr_{T\sim\mu_S}[T = \emptyset] = \mu_{\ell-1}(0)$. Formula (2) then shows that $\beta_{\ell+1} = \mu_\ell(0) = \ell\mu_{\ell-1}(0) - 1/(e-1)$. To recover our original definition, multiply everything by $1 - 1/e = (e-1)/e$.

5 Deterministic vs. randomized

Both the continuous algorithm and our algorithm are randomized. In constrast, the best deterministic algorithms known, greedy and local search, both have an approximation ratio of $1/2$. A similar situation arises in the case of unconstrained submodular maximization (for functions which are not necessarily monotone): the double greedy algorithm of Buchbinder, Feldman, Naor and Schwartz has a randomized variant giving the optimal $1/2$ approximation ratio, and a deterministic variant giving an approximation ratio of $1/3$, matching the approximation ratio of the deterministic local search algorithm of Feige, Mirrokni and Vondrák. Can it be the case that randomized algorithms outperform deterministic algorithms in the value oracle setting?