Filtered Search for Submodular Maximization with Controllable Approximation Bounds

Wenlin Chen, Yixin Chen, Kilian Q. Weinberger
Washington University in St. Louis
One Brookings Drive, St. Louis, MO 63130 USA
wenlinchen@wustl.edu, chen@cse.wustl.edu, kilian@wustl.edu

Abstract

Most existing submodular maximization algorithms provide theoretical guarantees with approximation bounds. However, in many cases, users may be interested in an anytime algorithm that can offer a flexible trade-off between computation time and optimality guarantees. In this paper, we propose a filtered search (FS) framework that allows the user to set an arbitrary approximation bound guarantee with a "tunable knob", from 0 (arbitrarily bad) to 1 (globally optimal). FS naturally handles monotone and non-monotone functions as well as unconstrained problems and problems with cardinality, matroid, and knapsack constraints. Further, it can also be applied to (non-negative) non-submodular functions and still gives controllable approximation bounds based on their submodularity ratio. Finally, FS encompasses the greedy algorithm as a special case. Our framework is based on theory in A^* search, but is substantially more efficient because it only requires heuristics that are critically admissible (CA) rather than admissible—a condition that gives more effective pruning and is substantially easier to implement.

1 Introduction

Submodular optimization has wide applications and its uses in machine learning are increasing. For example, it has been utilized for image segmentation (Jegelka and Bilmes, 2011; Kim et al., 2011; Jegelka et al., 2013), information retrieval (Yue and Guestrin, 2011; Lin and Bilmes, 2010), sensor placement (Krause and Guestrin, 2005; Krause et al., 2008), clustering (Narasimhan et al., 2005), speech recognition (Lin and Bilmes, 2009) and sparse methods (Bach, 2010; Das and Kempe, 2011).

A set function $g: 2^U \to \mathcal{R}$ defined on a finite set U is submodular if for all $S, T \subseteq U$ it satisfies $g(S)+g(T) \geq g(S \cap T)+g(S \cup T)$. In this paper, we focus on the following general problem:

$$\begin{aligned} \underset{S \subseteq U}{\text{maximize}} \quad & g(S), \\ \text{subject to} \quad & S \in \mathcal{F}, \end{aligned} \tag{1}$$

where $g(S)$ is *non-negative* and *submodular*, and $\mathcal{F} \subseteq 2^U$ denotes the set of feasible solutions. We assume throughout that $B \in \mathcal{F}$ implies $A \in \mathcal{F}$ for any $A \subseteq B \subseteq U$ and consequently the empty set is always feasible, *i.e.* $\emptyset \in \mathcal{F}$. We do not restrict $g(S)$ to be monotone. (A function is monotone if $g(S) \leq g(T)$ whenever $S \subseteq T$.) Let S^* be the optimal solution to (1) and $g^* = g(S^*)$ [1].

There has been extensive research on solving different cases of (1). For example, a greedy algorithm achieves a $(1-1/e)$-approximation bound for monotone functions with a cardinality constraint, *i.e.* $\mathcal{F} = \{S : |S| \leq K\}$, where K is a positive integer (Nemhauser et al., 1978). The greedy algorithm also gives a $\frac{1}{2}(1-1/e)$ bound for monotone submodular maximization under a knapsack constraint (Khuller et al., 1999), and a 1/2-approximation bound for monotone submodular maximization under a matroid constraint (Fisher et al., 1978). A multilinear extension method can give a $(1-1/e)$ bound for monotone submodular maximization under a matroid constraint (Calinescu et al., 2011). Various approximation bounds have also been studied for maximizing non-monotone submodular functions under different settings (Feige et al., 2011;

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

[1] We assume there is a unique S^* for simplicity. All results generalize to the case of non-unique solutions.

Lee et al., 2009; Vondrák et al., 2011).

Although the above algorithms are often efficient and provide approximation bounds, they have some limitations. For many important decision problems, it is vital to obtain high-quality solutions. People are willing to pay more computing time for better solution quality. At least, it is important to provide users with the flexibility to choose the trade-off between computing time and solution quality. Existing algorithms such as the greedy algorithm or deterministic local search are rigid and can only offer one fixed solution quality.

In this paper, we propose a filtered search (FS) framework that can support arbitrary α-approximation bounds for any $\alpha \in [0,1]$, providing a user with the flexibility to set his/her desired trade-off between solution quality and efficiency. One can consider the hyper-parameter α as a *tuning knob*. When $\alpha=0$, the search becomes the greedy algorithm; when $\alpha=1$, the search with a CA heuristic will find the global optimal solution.

Our framework utilizes a *state-space graph* for submodular optimization, in which each node is a subset and edges connect neighboring subsets. FS leverages classic theory for graph search, following the general structure of A^* search algorithms, but features a key innovation that exploits the special structure of submodular functions: For A^* search, the heuristic needs to be *admissible*. For submodular maximization this means that we need to compute an upper bound of $g^* - g(S)$ for every $S \subseteq U$. This admissibility condition is very strong, and may be hard to compute and typically requires long search time. We show that admissibility is not needed for optimal *submodular* maximization. Instead, we show that a sufficient condition to achieve optimality is *critical admissibility (CA)* of the heuristic function. The CA condition is much more effective than the admissibility condition and significantly reduces the search cost. It entails a novel "*optimal search with inadmissible heuristics*" approach which was just recently discovered for some planning problems (Karpas and Domshiak, 2012).

In many scenarios polynomial-time algorithms are inherently limited. For example, the (1-1/e) bound is optimal for monotone submodular maximization under a cardinality constraint in the value oracle model (Nemhauser and Wolsey, 1978). In contrast, our proposed FS approach can achieve arbitrary approximation bound, which means it will have exponential worst-case complexity in order to achieve higher bounds. However, in practice, FS is still useful since its exponential complexity is only for the worst case, and it may be more efficient than the theoretically efficient algorithms with high-order polynomial time complexity. This is akin to admissible search for AI, which optimally solves NP-hard graph search problems. Although its worst-case time complexity is exponential, it is efficient in many domains and remains the most widely used and most successful algorithm for many applications, such as game-playing, planning, and path finding. For example, heuristic search has been a leading method for automated planning, as manifested by the results of the recent International Planning Competitions (IPC).

In addition to its elegant efficiency/approximation tradeoff, FS also provides a general framework to solve various submodular maximization problems: it can handle both monotone and non-monotone functions; it naturally handles unconstrained problems and problems with cardinality, matroid, and knapsack constraints. Different constraints can be incorporated in a unified fashion – the user simply needs to provide an appropriate heuristic. In this paper, we derive efficient and tight CA heuristics for several important settings. Finally, the theoretical properties and advantages of the FS approach are backed by promising empirical results. We believe that FS will be a foundational and versatile framework for submodular optimization.

2 The Filtered Search (FS) Framework

We describe the general FS framework for solving (1). Define the *discrete derivative* for any $j \in U$ and $S \subseteq U$ as

$$g(j|S) \triangleq g(S \cup j) - g(S). \tag{2}$$

The discrete derivative gives rise to an alternative definition of submodularity: g is submodular if and only if it satisfies the *diminishing return* property, $g(j|S) \geq g(j|T)$ for all $S \subseteq T$ and $j \notin T$.

Definition 1 State-space graph. *For the submodular maximization problem in (1) the state-space graph is a directed graph $G = (\mathcal{F}, E)$, where the vertex set $\mathcal{F}$ is the set of all feasible subsets in U and the (directed) edge set E is such that for any two states $S, T \in \mathcal{F}$ there exists an edge $(S,T) \in E$ if and only if $S \subset T$ and $|T| = |S| + 1$.*

Although the state-space graph G may contain up to $2^{|U|}$ states, it is important to note that FS expands states in $\mathcal{F}$ on-demand and typically only examines a tiny fraction of $\mathcal{F}$. Similar to A^* search, instead of maximizing $g(S)$ directly, we define and maximize an auxiliary evaluation function $f(S)$.

Definition 2 Evaluation function. *For the submodular maximization problem in (1), with state-space*

Algorithm 1 Filtered search (FS) for submodular maximization

```
Input: α, G = (F, E), g, h
compute f(∅) = g(∅) + αh(∅)
MaxHeap.push(< ∅, f(∅) >)
ClosedList={}
while MaxHeap is not empty do
  S=MaxHeap.pop()  //S has the maximum f(S)
  if h(S) = 0 then
    return S     //solution found
  end if
  if S ∉ ClosedList then
    ClosedList=ClosedList ∪{S}
    for each (S, S') ∈ E do
      compute f(S') = g(S') + αh(S')
      MaxHeap.push(< S', f(S') >)
    end for
  end if
end while
```

graph $G = (\mathcal{F}, E)$, the evaluation function $f(S)$ on $S \in \mathcal{F}$ is defined as:

$$f(S) = g(S) + \alpha h(S), \tag{3}$$

where $0 \leq \alpha \leq 1$ is an ***approximation factor*** *and $h(S) : \mathcal{F} \to [0, +\infty)$ is a* ***heuristic function****.*

The FS framework is shown in Algorithm 1. It maintains two data structures, a max heap (which takes <value, key> pairs) and a closed list, and performs the following main steps:

0. Add the pair $< \emptyset, f(\emptyset) >$ to the max heap, where $\emptyset$ denotes the empty set.
1. Pop the state S from the heap with largest $f(S)$. If $h(S)=0$, return S as the solution.
2. For each edge $(S, S') \in E$ add $< S', f(S') >$ to the max heap if S' is not in the closed list.
3. Add S to the closed list and repeat from Step 1.

The closed list is implemented as a set with highly efficient hashing-based duplicate detection. As in most heuristic search procedures, the heuristic function is critical to the optimality and efficiency of the search algorithm.

2.1 Admissibility and Critical Admissibility

For graph search, the A^* search algorithm is optimal as long as the heuristic is admissible and consistent (Russell and Norvig, 2003). In our case we can define a corresponding definition of admissibility (which implies consistency, as in our setting the function $g(S)$ is independent on the path leading to S.)

Definition 3 Admissibility. *For the filtered search in Algorithm 1, a heuristic function $h(S) : \mathcal{F} \to [0, +\infty)$ is admissible if and only if for every state $S \in \mathcal{F}$, it satisfies that $h(S^*) = 0$ and*

$$h(S) \geq g^* - g(S). \tag{4}$$

In other words, admissibility guarantees that the heuristic $h(S)$ always *overestimates* the payoff to traverse from S to S^*.[2] A similar optimality result as in the case of A^* can be shown for our setup. The following theorem states that the solution for Algorithm 1 will be arbitrarily close to the optimal solution, depending on α and provided that the heuristic is admissible.

Theorem 1 *Algorithm 1 has an α-approximation bound whenever the heuristic function h is* admissible*, i.e., it returns a solution T that satisfies $g(T) \geq \alpha g^*$.*

The above result can be proved following similar reasoning for the optimality of A^* algorithm with admissible heuristics (Russell and Norvig, 2003). However, when the heuristic is admissible, even if it is almost perfect, it may take exponential search time[3] (Helmert and Röger). Moreover, in general it is difficult to ensure admissibility because g^* is hard to estimate. Thus, it is desirable to relax the strong admissibility conditions.

Fortunately, we can exploit the special structure of submodular maximization. We show that FS only requires a much weaker condition that is also much easier to enforce, which we refer to as *critical admissibility (CA)*. The CA condition only overestimates states on the optimal solution path and is defined as follows: 1. $h(S) = 0$ if there is no element $x \in U$ that can be added to S in order to increase the objective value without violating feasibility; 2. otherwise, we consider all the subsets $X \subseteq U$ that can be feasibly added to S and require $h(S)$ to be an upper bound on the function $\sum_{x \in X} g(x|S)$ (the cumulative increase in g if each element in X were to be added to S in isolation.)

Definition 4 Critical admissibility (CA). *For the filtered search in Algorithm 1, a heuristic function*

[2] In case of *minimization* problems admissibility is defined as a guarantee of *underestimation*.

[3] In the case of submodular maximization, an admissible heuristic with $\alpha = 1$ is useless because it simply overestimates every states and thus would become brute-force search. This is part of the reason why the CA heuristic is less strict as it is designed to overestimate states in the solution path (See Corollary 1).

$h(S) : \mathcal{F} \to [0, +\infty)$ is critically admissible if and only if for every state $S \in \mathcal{F}$, it satisfies

$$\begin{aligned} &1.\ h(S) = 0, \quad \text{if } g(x|S) \leq 0 \text{ or } S \cup \{x\} \notin \mathcal{F},\ \forall x \in U; \\ &2.\ h(S) \geq \max_{X \subseteq U : S \cup X \in \mathcal{F}} \sum_{x \in X} g(x|S), \quad \text{otherwise.} \end{aligned} \tag{5}$$

Remarks. CA has a couple of key advantages over admissibility. First, CA poses a less strict requirement as it does not require the heuristic to be admissible at every state. Consider a simple example where $U = \{1, 2, 3\}$ and $g(S) = \sum_{x \in S} x$ and $\mathcal{F}$ encodes a cardinality constraint $|S| \leq 2$. The optimal set is $S^* = \{2, 3\}$ with $g^* = 5$. Consider $S = \{1, 2\}$, which satisfies the first condition of CA and therefore has $h(S) = 0$. As $g^* - g(S) = 2$, we have that h is not admissible at S. It is however important to point out that for any $S \subset S^*$ a CA heuristic is admissible, in other words it does *overestimate* the payoff along possible solution paths. Second, CA is much easier to implement. It is hard to estimate g^* and thus hard to design an admissible heuristic function. In contrast, in the CA condition $\sum_{x \in X} g(x|S)$ is a modular function over X and its maximum can typically be computed very cheaply, depending on the constraint set $\mathcal{F}$. We will show concrete examples of CA heuristic functions for various scenarios in the following.

2.2 Optimality with CA Heuristics

We re-state the well-established result that a submodular function is bounded above by its modular approximation (Nemhauser et al., 1978) and then we present our main theorem and its proof for the approximation guarantee of Algorithm 1.

Lemma 1 *For any submodular function $g : 2^U \to \mathcal{R}$, we have $g(B) \leq g(A) + \sum_{s \in B - A} g(s|A)$, $\forall A \subseteq B \subseteq U$.*

Theorem 2 *Algorithm 1 has an α-approximation bound whenever the heuristic function h is critically admissible, i.e., it returns a solution T that satisfies $g(T) \geq \alpha g^*$.*

Proof: Let S^* be the optimal solution to (1). Suppose $T \neq S^*$, otherwise the theorem trivially holds. First note that at the beginning of the inner loop of Algorithm 1 (before line 6) there must be at least one set P on the max heap such that $P \subseteq S^*$. This holds trivially at the beginning with $P = \emptyset$. Whenever the set $P \subseteq S^*$ with maximum cardinality is popped from the max heap the algorithm either terminates (if $P = S^*$) or all its feasible successors that are not in the closed list are added to the max heap. By the assumption made on $\mathcal{F}$ earlier any subset of S^* is feasible, and there is at least one successor $P' \subseteq S^*$ with $|P'| = |P| + 1$, which is not yet on the closed list.

It therefore must be the case that right before T is popped from the max heap, there is a state $P \subseteq S^*$ in the max heap. As h is CA and $P \cup (S^* - P) \in \mathcal{F}$, we must have

$$h(P) \geq \max_{X \subseteq U :\ P \cup X \in \mathcal{F}} \sum_{x \in X} g(x|P) \geq \sum_{x \in S^* - P} g(x|P). \tag{6}$$

Thus, we have that

$$\begin{aligned} f(P) &= g(P) + \alpha h(P) \\ &\geq g(P) + \alpha \sum_{x \in S^* - P} g(x|P) \\ &\geq \alpha \left(g(P) + \sum_{x \in S^* - P} g(x|P) \right) \\ &\geq \alpha g(S^*) \end{aligned} \tag{7}$$

The first inequality follows from (6), the second inequality holds because $g(P)$ is non-negative and the third one follows from Lemma 1.

When T is popped it must have the largest f value in the max heap, thus $f(P) \leq f(T)$. On the other hand, since T is the solution, we have that $h(T) = 0$ (Line 7 of Algorithm 1). Therefore, we have:

$$f(P) \leq f(T) = g(T) + \alpha h(T) = g(T). \tag{8}$$

Combining (8) with (7), we obtain the result, $g(T) \geq f(P) \geq \alpha g(S^*)$. ■

Corollary 1 *The Evaluation function f with the CA heuristic and $\alpha = 1$ is guaranteed to overestimate subsets of the optimal solution,* i.e. *states in the optimal solution path.*

Proof: By setting $\alpha = 1$ in Eq. (7), we have that $f(P) \geq g(S^*)$ where $P \in S^*$. By Definition 4, we have $h(S^*) = 0$. Thus, $f(P) \geq g(S^*) + h(S^*) = f(S^*)$. ■

Note that Theorem 2 is quite general as our FS framework can achieve α-approximation bound with any submodular function (*e.g.* monotone or non-monotone) and any down-monotone constraint (*e.g.* matroid, knapsack and etc).

Special cases. There are some special cases under the FS framework. When $\alpha = 1$, the solution returned by FS is optimal. This provides us a systematic approach to optimally maximize submodular functions. The search can be sped up by successively tightening the heuristic function. Some other AI areas, such as

planning and robotics, greatly benefit from this approach.

When $\alpha = 0$, FS becomes the greedy algorithm, since at each state S the element s with the largest objective value, $g(s \cup S)$, is selected. If $g(S)$ is monotone, we get extra optimality guarantees: an $(1-1/e)$-approximation bound for the cardinality constraint, and a $1/(p+1)$ bound for p matroid constraints (Fisher et al., 1978).

When $0 < \alpha < 1$, α gives direct control over the trade-off between solution quality and speed. A small α favors speed over quality while a large α offers a high approximation bound at the cost of a more expensive search. FS can be turned into an *anytime* algorithm by starting with $\alpha = 0$ and repeatedly resolving the problem with slightly increased α. When interrupted, the solution with the highest objective value is returned.

2.3 Non-submodular Function Maximization

FS can also be applied to non-submodular functions and we can still provide approximation guarantees, using the notion of submodularity ratio (Das and Kempe, 2011; Grubb and Bagnell, 2012).

Definition 5 Submodularity Ratio. *Given a ground set U and feasible set $\mathcal{F}$, a non-negative set function g has the submodularity ratio γ if*

$$\sum_{x \in X} g(x|L) \geq \gamma \left[g(L \cup X) - g(L) \right] \tag{9}$$

for all $L, X \subseteq U$ such that $X \cup L \in \mathcal{F}$ and $X \cap L = \emptyset$.

The submodularity ratio, γ, characterizes how close to submodular a set function is ($\gamma = 1$ means submodular and $0 \leq \gamma < 1$ means non-submodular)[4].

Theorem 3 *For a set function g with a submodularity ratio of γ, Algorithm 1 returns a solution T with an $\alpha\gamma$-approximation bound.*

Proof: The proof of Theorem 2 can be directly applied to this theorem except that the last inequality in (7) does not hold anymore since g is not submodular. But with the definition of submodularity ratio in (9), we can modify the proof of Theorem 2 and start off right before the last inequality in (7):

$$\alpha \Big(g(P) + \sum_{x \in S^* - P} g(x|P) \Big) \tag{10}$$

$$\geq \alpha \Big(g(P) + \gamma g(S^*) - \gamma g(P) \Big) \tag{11}$$

$$\geq \alpha\gamma g(S^*) + \alpha(1-\gamma) g(P) \tag{12}$$

$$\geq \alpha\gamma g(S^*) \tag{13}$$

(11) holds by the definition of submodularity ratio with $L = P$ and $X = S^* - P$. Following the same reasoning as before, we obtain $g(T) \geq \alpha\gamma g(S^*)$. ■

3 Critically Admissible Heuristics

In this section, we develop CA heuristics for several important scenarios of submodular maximization, such as matroid, cardinality (included as a special case of matroid), and knapsack constraints. The proposed heuristics are tight (except for knapsack constraints) and general, since they assume a value oracle model where the set function g is a blackbox and accessible only via evaluation.

Matroid constraints. A *matroid* is denoted as a pair $(U, \mathcal{I})$ where U is a finite ground set and $\mathcal{I} \subseteq 2^U$ is a set of subsets of U satisfying the following two properties: (1) If $Y \in \mathcal{I}$ and $X \subseteq Y$, then $X \in \mathcal{I}$. (2) If $X, Y \in \mathcal{I}$ and $|X| < |Y|$, then there exists some $u \in Y - X$ such that $X \cup \{u\} \in \mathcal{I}$.

A special case of the matroid constraint is the **cardinality constraint**, where $(U, \mathcal{I})$ is a *uniform* matroid: $\mathcal{I} = \{X \subseteq U : |X| \leq K\}$. Another example is the *partition* matroid in which U is partitioned into disjoint sets $U_1, U_2, \cdots, U_m$, and $\mathcal{I} = \{X \subseteq U : |X \cap U_i| \leq K_i \text{ for } i = 1, \cdots, m\}$. Submodular maximization with a matroid constraint is formulated as:

$$\underset{S \subseteq U}{\text{maximize}}\ g(S) \qquad \text{subject to } S \in \mathcal{I}. \tag{14}$$

We will now provide a CA heuristic for the case $\mathcal{F} = \mathcal{I}$ that is **tight** for every state S, *i.e.* every $h(S)$ satisfies the CA conditions (5) with *equalities*. The first case in (5) can easily be satisfied by setting $h(S) = 0$ whenever the condition is met. We therefore consider the second case and show that we can compute $h(S) = \max_{X \subseteq U : S \cup X \in \mathcal{F}} \sum_{x \in X} g(x|S)$ with a simple, efficient greedy algorithm. We iteratively construct a solution set X^T, starting from an empty set $X^0 = \emptyset$. In each iteration we pick the element x^t that satisfies $X^{t-1} \cup \{x^t\} \in \mathcal{I}$ and maximizes $g(x^t|S)$ and add it to our set, $X^t = X^{t-1} \cup \{x^t\}$. We stop when no

[4] Strictly speaking, the original definition of submodularity ratio requires $|X| \leq K$ where K is the cardinality constraint. We generalize it to $X \cup L \in \mathcal{F}$ which is stricter and shares all the properties of the original definition, as proposed in (Das and Kempe, 2011).

more element x^t can be found or $g(x^t|S)<0$. Because we are maximizing a *modular* function, the solution X^T returned by this greedy approach is provably optimal (Calinescu et al., 2011; Nemhauser et al., 1978). Using Theorem 2, we have the following corollary:

Corollary 2 *The solution of Algorithm 1 with the above heuristic function $h(S)$ guarantees an α-approximation bound for the submodular function maximization with a matroid constraint* (14)*.*

Knapsack constraints. We also consider submodular maximization with a knapsack constraint, defined as:

$$\max_{S\subseteq U} g(S) \quad s.t. \sum_{x\in S} c_x \leq B, \tag{15}$$

where $c_x \geq 0$ is the cost of x and $B \geq 0$ is the budget limit. Here, the second CA condition in (5) is:

$$\begin{aligned} h(S) \geq & \max_{X:X\subseteq U-S} \sum_{x\in X} g(x|S), \\ s.t. & \sum_{x\in X} c_x \leq B'_S, \end{aligned} \tag{16}$$

where $B'_S = B - \sum_{s\in S} c_s$ is the remaining budget for state S. Different from the case of matroid constraints, the equality case of (16) describes a *0-1 knapsack* problem (Cormen et al., 2001) and is NP-hard. However, we can upper bound it by the solution of the corresponding *fractional knapsack* problem (Cormen et al., 2001):

$$\begin{aligned} h(S) = & \max_{0\leq w_x\leq 1} \sum_{x\in U-S} w_x g(x|S) \\ s.t. & \sum_{x\in U-S} w_x c_x \leq B'_S. \end{aligned} \tag{17}$$

This relaxed version can be solved efficiently with a greedy algorithm (Cormen et al., 2001), leading to the heuristic

$$h(S) = \sum_{i=1}^{j} g(x_i|S) + \frac{1}{c_{x_{j+1}}}\left(B'_S - \sum_{i=1}^{j} c_{x_i}\right) g(x_{j+1}|S), \tag{18}$$

where $x_i \in U$ are sorted in decreasing order of $g(x_i|S)/c_{x_i}$ and j is the lowest index such that $\sum_{i=1}^{j} g(x_i|S) \leq B'_S$. From Theorem 2, we have the following corollary:

Corollary 3 *The solution of Algorithm 1 with the heuristic $h(S)$ defined in* (18) *guarantees an α-approximation bound for the submodular function maximization problem with a knapsack constraint,* (15)*.*

The above CA heuristic can be extended to the case of multiple knapsack constraints. We can still compute the heuristic by relaxing the original combinatorial knapsack optimization into a continuous fractional knapsack optimization, which is a linear programming (LP) problem and can be solved very efficiently with LP solvers such as IBM CPLEX.

Relaxed critical admissibility. We have shown that in the case of a single matroid or knapsack constraint, the CA conditions in (5) can be achieved in a straight-forward manner because of its modularity. However, for some problems, *e.g. multiple* matroid constraints, solving (5) tightly induces high time complexity. In these cases, we can relax the CA condition and still obtain an optimality guarantee for FS.

Theorem 4 *Suppose the heuristic $h(S)$ has a β-approximation bound for* (5)*,* i.e. *$h(S) \geq \beta h^*(S)$ where $h^*(S)$ is a tight CA heuristic and $0 \leq \beta \leq 1$, then the solution returned by Algorithm 1 achieves: a) an α-approximation bound with the heuristic $\frac{h(S)}{\beta}$; and b) an $\alpha\beta$-approximation bound with the heuristic $h(S)$.*

Proof: a) holds because $\frac{h(S)}{\beta} \geq h^*(S)$ satisfies CA and thus Theorem 2 applies. For b), the proof is the same as Theorem 2 except that (6) should be replaced by $h(P) \geq \beta \sum_{x\in S^*-P} g(x|P)$, and α replaced by $\alpha\beta$ after the last two inequalities in (7). ■

We can apply this theorem to submodular maximization with p matroid constraints where the feasible solution set $\mathcal{F}$ is defined by p matroids $\mathcal{I}_1 \cap \cdots \cap \mathcal{I}_p$. Suppose $h(S)$ is the objective value returned by the greedy algorithm described in Section 3. It is shown that $h(S)$ has an $\frac{1}{p}$-approximation bound for (5) (Calinescu et al., 2011; Fisher et al., 1978), which leads to our final corollary:

Corollary 4 *For* (1) *with p matroid constraints, Algorithm 1 guarantees an $\frac{\alpha}{p}$-approximation bound with heuristic $h(S)$ and an α-approximation bound with heuristic $ph(S)$.*

4 Related Work

In the introduction we have already reviewed some algorithms to solve Eq. (1) with constraints, and approximation bounds for non-monotone submodular functions. Here, we provide some more detail on the second scenario. For unconstrained problems, Feige et al. (2011) show that a deterministic local search algorithm achieves a $(\frac{1}{3} - \frac{\epsilon}{n})$-approximation ratio with at most $O(\frac{1}{\epsilon} n^3 \log n)$ oracle calls. A similar local search procedure with additional exchange operations obtains $\frac{1}{(1+\epsilon)(k+2+1/k)}$-approximation over k matroids (Lee et al., 2009). Vondrák et al. (2011) propose multilinear relaxation and contention resolution schemes

to achieve 0.325-approximation for a constant number of knapsack constraints and $0.19/k$-approximation for additional k matroid constraints.

There have been some works concerning the exact maximization of submodular function leveraging branch and bound (BB) approach. However, most of them have strong assumptions on the problems to be solved. The submodularity cut proposed in (Kawahara et al., 2009) is limited to cardinality constraints and their "epsilon-optimality" only holds for non-decreasing functions. The BB approach presented in (Goldengorin, 2009; Goldengorin et al., 1999) can only handle unconstrained problems. Nemhauser and Wolsey (1981) also propose a BB approach to maximizing non-decreasing submodular functions. For general submodular functions, it is limited to linear constraints. In contrast, our framework is general enough to handle any down-monotone constraints and non-monotone functions, which is not found in any of these prior works.

Recently, Iyer et al. (2013) proposed an elegant discrete semidifferential-based framework for general submodular maximization and minimization. The key idea is to approximate the submodular function with a tight modular lower (or upper) bound, which is easier to optimize. The same authors also utilize a similar idea to deal with submodular cover and knapsack constraints (Iyer and Bilmes, 2013). In contrast to our FS approach, these methods still feature specific approximation bounds.

Karpas and Domshiak (2012) propose the "optimal search with inadmissible heuristics" paradigm, which follows a similar theme as FS: in general A^* search needs admissible heuristics to guarantee optimality, but for specific types of problems we might exploit their structure to relax the admissibility condition without compromising optimality. Their focus is on automated planning problems, where they introduce a weaker condition, globally admissibility (GA), which can be used to replace admissibility. Although similar in spirit, our framework is quite different and focuses on submodular optimization problems, where we explicitly exploit the submodular properties of the objective.

5 Experimental Results

In this section we conduct experiments to empirically evaluate the proposed FS framework.

Our experiments are based on a speech recognition task which selects a subset from an un-transcribed corpus of speech utterance for transcription (Lin and Bilmes, 2009). The original optimization is maximizing a cut function $g(S) = \sum_{i\in U}\sum_{j\in S} w_{ij} - \sum_{i,j\in S} w_{ij}$ under a cardinality constraint where w_{ij} is the similarity between utterances. The same cut function with a knapsack constraint is also widely used in documentation summarization tasks (Lin and Bilmes, 2010). The cut function is submodular and non-monotone. In our experiments, the similarity matrix is computed based on the TIMIT corpus (Garofolo et al., 1993) using a string kernel metric (Lin and Bilmes, 2009). For completeness, we evaluate the performance of maximizing a cut function subject to a cardinality constraint, a partition matroid constraint and a knapsack constraint, respectively. Following the settings of (Iyer et al., 2013), we test each setup with 100 sampled similarity matrices, each of size $20 \leq |U| \leq 30$ so that its optimal solution can still be achieved with $\alpha = 1$ in the FS framework. For the cardinality constrained problem, we set the cardinality limit to 10. For the partition matroid constrained problem, there are 5 random partitions of size 2. For knapsack constraints, the cost for each item is randomly generated and the budget limit is $\frac{1}{4}$ of the total cost.

For comparison, we also implement several baseline methods most of which have been tested on this particular cut function. We have the greedy algorithm (Greedy) (Nemhauser et al., 1978; Fisher et al., 1978) and Lee's algorithm (Lee) (Lee et al., 2009) for matroid constrained problems (including cardinality and partition matroid constraints). For knapsack constrained problems, we test the cost-aware greedy algorithm (CGreedy) (Khuller et al., 1999; Lin and Bilmes, 2010), and the improved greedy algorithm (N3Greedy) (Khuller et al., 1999; Sviridenko, 2004) which finds all greedy solutions starting from any set of size 3. Note that all these methods except Lee's are encompassed by the MMax framework (Iyer et al., 2013). In Figure 1, we show the objective values (averaged over all randomized runs) of the baseline algorithms and FS, with α ranging from 0 to 1.

We make several observations: 1. as predicted, the performance of FS with $\alpha = 0$ matches the greedy algorithm exactly for problems with a cardinality or partition matroid constraint; 2. the objective values of the FS solutions tend to increase with growing α—an encouraging observation, as the α-approximation is for the worst case and in practice does not guarantee monotonically increasing objective values for increasing α. 3. FS tends to obtain reliably better solutions than competing methods already with $\alpha = 0.5$; 4. the graphs in the bottom row show that the improvement of solution quality comes at a price of increased time complexity, especially as $\alpha \to 1$.

For experiments shown in Figure 1(a) and (b), the sudden increase of running time happens because FS

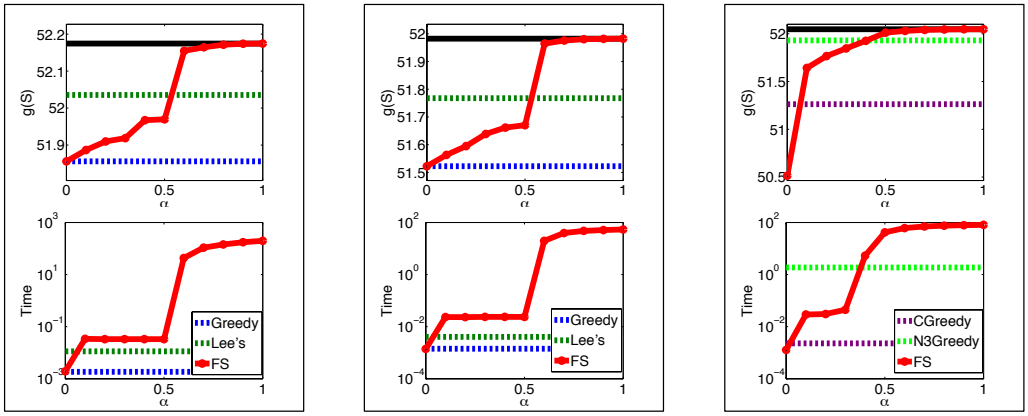

Figure 1: The (averaged) objective value (*upper* plots) and running time in seconds (*lower* plots) as a function of α. The dashed lines represent the various baselines and the black lines the global optimum.

induces exponentially increasing backtracking at that point. The curve is generally smoother for the knapsack constraints as in Figure 1(c). The "smoothness" of this curve depends on the problem structure including the properties of the objective function and the constraints, which requires further research and is likely to inspire more variants of FS.

Worst case for greedy algorithm. Note that although the greedy algorithm can perform well in practice (as shown in Figure 1), its theoretical approximation guarantee is pretty loose. For sanity check, we conduct the experiment specified in (Pan et al., appendix) where the performance of greedy algorithm can be arbitrarily bad depending on the cardinality of the ground set N. In particular, the *empirical* approximation bound for greedy algorithm is always $\frac{1}{N}$. In contrast, FS can obtain the optimal solution even with small α (*e.g.* $\alpha = 0.1$).

6 Conclusion

In summary, Filtered Search (FS) provides a unified framework for solving submodular optimization problems with various constraints, regardless of function monotonicity or the constraint type. For non-submodular functions it also provides an approximation guarantee, based on their submodularity ratio. In addition to the strong theoretical results, we also demonstrated empirically that FS is a practical algorithm that shows superior results in several real world settings. Our experimental findings further suggest that users can control the trade-off between solution quality and efficiency, which parallels our theoretical contribution of a variable approximation guarantee.

FS leverages existing theory in A^* search, but is substantially more efficient as it only requires heuristics that are critically admissible (CA) rather than admissible—a novel condition that is much more relaxed and substantially easier to implement. We developed several efficient CA heuristics for various constraint types under the value-oracle model and hope that these heuristics can serve as templates for future problem settings.

Our work bridges submodular optimization with graph search for AI, and can potentially benefit from the extensive existing research in both areas. For example, we plan to integrate space-pruning techniques, such as partial-order reduction and symmetry detection (Chen and Yao, 2009), into FS. We believe that FS will be a foundational and unifying framework for submodular optimization, and will foster cross-fertilization of the two fields.

Acknowledgements

W. Chen and Y. Chen were partially supported by the CNS-1017701, CCF-1215302, IIS-1343896, and DBI-1356669 grants from the National Science Foundation of the US. K.Q. Weinberger was supported by NSF grants IIA-1355406, IIS-1149882, EFRI-1137211.

References

The International Planning Competitions. *http://ipc.icaps-conference.org/*.

F. Bach. Structured sparsity-inducing norms through submodular functions. In *NIPS*, 2010.

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

Y. Chen and G. Yao. Completeness and Optimality Preserving Reduction for Planning. In *IJCAI*, 2009.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, 2011.

U. Feige, V. S. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

M. L Fisher, G. L Nemhauser, and L. A Wolsey. An analysis of approximations for maximizing submodular set functions II. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and Z. Victor. Timit: acoustic-phonetic continuous speech corpus. In *DARPA*, 1993.

B. Goldengorin. Maximization of submodular functions: Theory and enumeration algorithms. *European Journal of Operational Research*, 198(1):102–112, 2009.

B. Goldengorin, G. Sierksma, G. A. Tijssen, and M. Tso. The data-correcting algorithm for the minimization of supermodular functions. *Management Science*, 45(11):1539–1551, 1999.

A. Grubb and D. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *AISTATS*, 2012.

M. Helmert and G. Röger. How good is almost perfect. In *AAAI'08*.

R. Iyer and J. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*, 2013.

R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML*, 2013.

S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, 2011.

S. Jegelka, F. Bach, and S. Sra. Reflection methods for user-friendly submodular optimization. In *NIPS*, 2013.

E. Karpas and C. Domshiak. Optimal search with inadmissible heuristics. In *ICAPS*, 2012.

Yoshinobu Kawahara, Kiyohito Nagano, Koji Tsuda, and Jeff Bilmes. Submodularity cuts and applications. In *NIPS*, 2009.

S. Khuller, A. Moss, and J. S. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

G. Kim, E. P. Xing, F. Li, and T. Kanade. Distributed cosegmentation via submodular optimization on anisotropic diffusion. In *ICCV*, 2011.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*, 2009.

H. Lin and J. Bilmes. How to select a good training-data subset for transcription: Submodular active selection for sequences. In *INTERSPEECH*, 2009.

H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. ACL-HLT, 2010.

M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. In *NIPS*, 2005.

G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3 (3):177–188, 1978.

G L Nemhauser and L A Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *North-Holland Mathematics Studies*, 59: 279–301, 1981.

G. L. Nemhauser, L. A Wolsey, and M. L Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14 (1):265–294, 1978.

X. Pan, S. Jegelka, J. Gonzalez, J. Bradley, and M. Jordan. Parallel double greedy submodular maximization. In *NIPS'14*.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Second edition, 2003.

M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

J. Vondrák, C. Chekuri, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *STOC*, 2011.

Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *NIPS*. 2011.