CSC2420 - Algorithm Design, Analysis and Theory **Fall 2022**

Lecture 5: Matroid, Local Search

Lecturer: Allan Borodin *Scribe: Kevin Gao*

5.1 Set Packing

We will consider the s-set packing problem and two greedy algorithms: one relatively "natural" greedy algorithm, and one that is not so natural.

Problem 5.1. *We are given n subsets $S_1, \ldots, S_n$ from a universe U of size m. In the weighted case, each subset S_i has a weight w_i. The goal is to choose a disjoint subcollection $\mathcal{S}$ of the subsets so as to maximize $\sum_{S_i \in \mathcal{S}} w_i$. In the s-set packing problem, we have $|S_i| \leq s$ for all i.*

By reduction from max-clique, there is an $m^{1/2-\epsilon}$ hardness of approximation result assuming NP $\neq$ ZPP. For s-set packing with constant $s \geq 3$, there is an $\Omega(s/\log s)$ hardness of approximation assuming P $\neq$ NP.

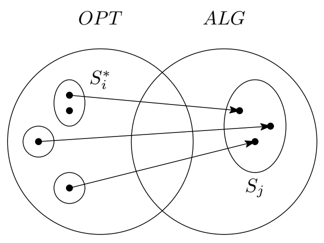
5.1.1 Greedy by Weight

```
Greedy-By-Weight
1  sort the sets so that $w_1 \geq w_2 \geq \ldots \geq w_n$
2  $\mathcal{S} = \emptyset$
3  for $i = 1$ to $n$
4      if $S_i$ does not intersect any set in $\mathcal{S}$
5          $\mathcal{S} = \mathcal{S} \cup \{S_i\}$
```

Theorem 5.2. Greedy-By-Weight *provides an s-approximation for the s-set packing problem.*

Proof:

We prove this using a charging argument. It suffices to show a charging function $\chi : OPT \to ALG$ such that for any set $S_j \in ALG$, the weight of sets in OPT charged to S_j is at most $s \cdot w_j$. We charge the weight of every set in OPT to the first set in the greedy solution with which it intersects. Such set must exist in ALG because otherwise if some set in OPT does not intersect with any sets in ALG, it can just be taken by ALG as part of the solution entirely. Since $|S_j| \geq s$, at most s sets in OPT can be charged to the same S_j. If S_i^* is charged to S_j, then $w_i \leq w_j$ because otherwise greedy would have taken S_i^* instead.



■

5.1.2 Charging Argument

The argument that we used for Theorem 5.2 is known as the charging argument. It can be used to show both the correctness (optimality) of greedy algorithms and the approximation ratio. To show some algorithm obtains the globally optimal solution of some maximization problem, one needs to show that the algorithm produces an output with as much "profit" as the optimal solution. It suffices to find a 1-to-1 function $f : OPT \to ALG$. By providing such a function and showing that it is 1-to-1, we would have $|OPT| \le |ALG|$, which proves the optimality of the algorithm. Similarly, for minimization problem, it suffices to give a 1-to-1 function $h : ALG \to OPT$ which would imply $|ALG| \le |OPT|$.

The same argument can be used to prove approximation ratios. For maximization problem, we charge (map) the profit of some optimal solution OPT to the profit of ALG. We can then argue that not too much profit from OPT gets charged (mapped) to ALG. For this, it suffices to prove that the charging function is ϵ-to-1 where ϵ is the desired approximation ratio.

5.1.3 Greedy by Weight per Square Root of Size

If all sets have unit weight, greedy by weight-per-size is known to provide a $\sqrt{s}$-approximation, but this is just the same as ordering the input sets so that $|S_1| \le |S_2| \le \ldots \le |S_n|$. However, greedy by weight-per-size does not improve the s approximation ratio for the weighted case. One heuristic that improves the approximation ratio is to instead order the sets by their weight-per-square root of size.

```
GREEDY-BY-WEIGHT-PER-SQRT-SIZE
1  sort the sets so that $\frac{w_1}{\sqrt{|S_1|}} \ge \frac{w_2}{\sqrt{|S_2}} \ge \ldots \ge \frac{w_n}{\sqrt{|S_n|}}$
2  $\mathcal{S} = \emptyset$
3  for $i = 1$ to $n$
4      if $S_i$ does not intersect any set in $\mathcal{S}$
5          $\mathcal{S} = \mathcal{S} \cup \{S_i\}$
```

Theorem 5.3. GREEDY-BY-WEIGHT-PER-SQRT-SIZE *provides a $2\sqrt{m}$-approximation for the set packing algorithm. We note that $\sqrt{m}$ is asymptotically the best possible approximation assuming NP $\neq$ ZPP.*

5.1.4 Partial Enumeration Greedy

Alternatively, one can use the partial enumeration greedy approach for the set packing problem. The algorithm PARTIAL-GREEDY is described as follows. Let MAX_k be the optimal solution of the set packing problem restricted to those containing at most k sets. Let G be the solution by GREEDY-BY-WEIGHT restricted to the sets of cardinality of at most $\sqrt{m/k}$. Return the best of MAX_k and G.

Theorem 5.4. PARTIAL-GREEDY *achieves a $2\sqrt{m/k}$ approximation for the weighted set packing problem on a universe of size m.*

5.2 Matroids

Definition 5.5 (Matroid)**.** *Let U be a set of elements and $\mathcal{I}$ be a collection of subsets of U. $(U,\mathcal{I})$ is a **matroid** if the following holds:*

- *heraitary property: if $I \in \mathcal{I}$ and $I' \subset I$, then $I' \in \mathcal{I}$*
- *exchange property: if $I', I \in \mathcal{I}$ and $|I'| < |I|$, then $\exists u \in I \setminus I'. I' \cup \{u\} \in \mathcal{I}$*

An **hereditary set system** $(U, \mathcal{I})$ is any set system satisfying the hereditary property so that a matroid is a hereditary set system that also satisfies the exchange property. The sets $I \in \mathcal{I}$ are called the **independent sets**. The **maximal independent set** of a matroid is also referred to as the **bases** of the matroid. By the exchange property, all maximal independent sets of a given matroid have the same cardinality, known as the **rank** of the matroid. Many of the terminologies used for matroid are similar to those in linear algebra. This is best illustrated with a **graph matroid** and the corresponding **vector matroid** induced by the graph.

Alternatively, a matroid can be defined in terms of **circuits**.

Definition 5.6 (Alternative Definition of Matriod)**.** *Let U be a set of elements and $\mathcal{C}$ be a collection of subsets of U. $(U, \mathcal{C})$ is a matroid if*

- *If $C_1, C_2 \in \mathcal{C}$ and $C_1 \subseteq C_2$, then $C_1 = C_2$*
- *If C_1 and C_2 are distinct members of $\mathcal{C}$ and $e \in C_1 \cap C_2$, then $\mathcal{C}$ contains some C_3 such that $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.*

The objective of max weighted independent set in a matroid is to select an independent set $I \subseteq U$ that maximizes $\sum_{u \in I} w(u)$ where $w(u)$ is the weight of element u.

Kruskal's algorithm for MST can be thought as the max weighted independent set problem on a graph matroid. For example, an independent set of vectors in a vector space is a matroid (Whitney 1935).

A standard greedy algorithm for a max independent set.

```
1  $\mathcal{S} = \emptyset$
2  **while** $\exists u. \mathcal{S} \cup \{u\}$ is independent
3      $\mathcal{S} = \mathcal{S} \cup \{u\}$ where $u$ is an element of largest weight such that $\mathcal{S} \cup \{u\}$ is independent
```

Theorem 5.7 (Rados)**.** *For independence in a matroid M, Greedy is an optimal algorithm for maximizing the weight of basis (i.e. maximizing a linear function subject to a matroid constraint).*

Theorem 5.8 (Edmonds)**.** *Consider any hereditary set system. If the natural greedy algorithm produces an optimal solution for every linear function of elements in an independent set, then the set system is a greedy algorithm.*

Does matroid formalizes greedy algorithms? No, it does not say anything about approximation algorithms.

Is the optimality of greedy algorithms limited to independent sets in matroids? No. Consider the greedy algorithm for unweighted interval scheduling on m identical machines, which does not satisfy the exchange property.

5.3 Local Search

Along with greedy and greedy-like algorithms, local search is one of the two conceptually simplest optimization paradigms.

```
1  initialize $S$
2  **while** there is a better solution $S'$ in the local nbhd of $S$
3      $S = S'$
```

If and when the algorithm terminates, the algorithm has computed a local optimum. However, some key problems remain:

- How are we allowed to choose an initial solution?
- What constitutes a reasonable definition of a "local neighborhood"?
- What do we mean by "better"?

For some applications, it turns out that rather than searching to improve the given objective function, we search for a solution in the local neighborhood that improves a related **potential function** and this has been termed **non-oblivious local search**.

5.3.1 Oblivious Local Search for Exact Max-2-SAT

Recall the Max-k-SAT problem. We are given a CNF formula with k variables. We would like to find a truth assignment that maximizes the number of satisfied clauses. In the weighted case, we want to instead maximize the combined weight of the satisfied clauses. The unweighted case is simply the weighted case with the weight of each clause set to 1.

```
MAX-2-SAT-LOCAL-SEARCH
1  $\tau$ = any initial truth assignment
2  while $\exists\hat{\tau} \in N_d(\tau)$, $W(\hat{\tau}) > W(\tau)$
3        $\tau = \hat{\tau}$
```

where $N_d(\tau)$ represents the d-neighborhood of τ defined to be $\{\tau' \mid \tau \text{ and } \tau' \text{ differ on at most } d \text{ variables}\}$.

Theorem 5.9. *The approximation ratio for* MAX-2-SAT-LOCAL-SEARCH *is* 2/3.

Proof: Let τ be a local optimum and let

- S_0 be the set of clauses not satisfied by τ
- S_1 be the set of clauses satisfied with exactly one literal by τ
- S_2 be the set of clauses satisfied with two literals by τ

Let $W(S_i)$ be the corresponding weight. We say a clause involves a variable x_j if either x_j or $\neg x_j$ occurs in the clause. Then, for each j, let

- A_j be those clauses in S_0 involving the variable x_j.
- B_j be those clauses C in S_1 involving the variable x_j such that it is the literal x_j or $\neg x_j$ that is satisfied in C by τ.
- C_j be those clauses in S_2 involving x_j.

Similarly, let $W(A_j), W(B_j), W(C_j)$ be the corresponding weights. Summing the weights over all variables, we get

$$2W(S_0) = \sum_j W(A_j)$$

becuase each clause in S_0 gets counted twice when we sum over the variables. For S_1, we have

$$W(S_1) = \sum_j W(B_j).$$

Given that τ is a local optimum, for every j, we have

$$W(A_j) \leq W(B_j)$$

or else flipping the truth value of x_j would improve the weight of the clauses being satisfied. Hence, by summing over all j, we have $2W_0 \leq W_1$. It follows that the ratio of clause weights not satisfied to the numb of all clause weights is

$$\frac{W(S_0)}{W(S_0)+W(S_1)+W(S_2)} \leq \frac{W(S_0)}{3W(S_0)+W(S_2)} \leq \frac{W(S_0)}{3W(S_0)}.$$

■

From the analysis of the approximation ratio of the oblivious local search algorithm, we can gain some knowledge as to how to improve the algorithm. Note that we are not using anything about $W(S_2)$ in the analysis. If we could guarantee that $W(S_0)$ was at most $W(S_2)$, then the ratio of clause weights not satisfied to all clause weights would be 1/4. The idea of the improved non-oblivious local search algorithm is to weigh S_2 clauses more heavily. Intuitively, the S_2 clauses are more valuable since they are more "resilient" to any single variable change.

Specifically, in the improved algorithm for Max-2-SAT, in each iteration where we find $\tau' \in N_1(\tau)$, we take the one that improves the **potential function**

$$\frac{3}{2}W(S_1) + 2W(S_2)$$

instead of $W(S_1) + W(S_2)$.

More generally, for Max-k-SAT, we set a set of scaling coefficients $c_1, \ldots, c_k$ such that the non-oblivious local search optimizes the potential function

$$c_1W(S_1) + c_2W(S_2) + \cdots + c_kW(S_k).$$

This gives an approximation ratio of $\frac{2^k-1}{2^k}$ for exact Max-k-SAT.

Next, we will sketch a proof for the 3/4 approximation ratio for the non-oblivious local search algorithm for Max-2-SAT.

Theorem 5.10. Non-Oblivious-Max-2-SAT *achieves an approximation ratio of* 3/4.

Proof: Without loss of generality, we assume that τ is the all-true assignment. Let $P_{i,j}$ be the weight of all clauses in S_i containing x_j. Let $N_{i,j}$ be the weight of all clauses in S_i containing $\neg x_j$. For some local optimum $\hat{\tau}$ with respect to the potential function $3/2W(S_1) + 2W(S_2)$, for all variables x_j,

$$-\frac{1}{2}P_{2,j} - \frac{3}{2}P_{1,j} + \frac{1}{2}N_{1,j} + \frac{3}{2}N_{0,j} \leq 0.$$

Summing over all variables, $P_1 = N_1 = W(S_1)$ and $P_2 = 2W(S_2)$ and $N_0 = 2W(S_0)$. It follows from the above inequality that

$$3W(S_0) \leq W(S_1) + W(S_2).$$

■