★ 16.4 Theoretical foundations for greedy methods

There is a beautiful theory about greedy algorithms, which we sketch in this section. This theory is useful in determining when the greedy method yields optimal solutions. It involves combinatorial structures known as "matroids." Although this theory does not cover all cases for which a greedy method applies (for example, it does not cover the activity-selection problem of Section 16.1 or the Huffman coding problem of Section 16.3), it does cover many cases of practical interest. Furthermore, this theory is being rapidly developed and extended to cover many more applications; see the notes at the end of this chapter for references.

Matroids

A ***matroid*** is an ordered pair $M = (S, \mathcal{I})$ satisfying the following conditions.

1. S is a finite nonempty set.
2. $\mathcal{I}$ is a nonempty family of subsets of S, called the ***independent*** subsets of S, such that if $B \in \mathcal{I}$ and $A \subseteq B$, then $A \in \mathcal{I}$. We say that $\mathcal{I}$ is ***hereditary*** if it satisfies this property. Note that the empty set $\emptyset$ is necessarily a member of $\mathcal{I}$.
3. If $A \in \mathcal{I}$, $B \in \mathcal{I}$, and $|A| < |B|$, then there is some element $x \in B - A$ such that $A \cup \{x\} \in \mathcal{I}$. We say that M satisfies the ***exchange property***.

The word "matroid" is due to Hassler Whitney. He was studying ***matric matroids***, in which the elements of S are the rows of a given matrix and a set of rows is independent if they are linearly independent in the usual sense. It is easy to show that this structure defines a matroid (see Exercise 16.4-2).

As another example of matroids, consider the ***graphic matroid*** $M_G = (S_G, \mathcal{I}_G)$ defined in terms of a given undirected graph $G = (V, E)$ as follows.

- The set S_G is defined to be E, the set of edges of G.
- If A is a subset of E, then $A \in \mathcal{I}_G$ if and only if A is acyclic. That is, a set of edges A is independent if and only if the subgraph $G_A = (V, A)$ forms a forest.

The graphic matroid M_G is closely related to the minimum-spanning-tree problem, which is covered in detail in Chapter 23.

Theorem 16.5
If $G = (V, E)$ is an undirected graph, then $M_G = (S_G, \mathcal{I}_G)$ is a matroid.

Proof Clearly, $S_G = E$ is a finite set. Furthermore, $\mathcal{I}_G$ is hereditary, since a subset of a forest is a forest. Putting it another way, removing edges from an acyclic set of edges cannot create cycles.

for any $A \subseteq S$. For example, if we let $w(e)$ denote the length of an edge e in a graphic matroid M_G, then $w(A)$ is the total length of the edges in edge set A.

Greedy algorithms on a weighted matroid

Many problems for which a greedy approach provides optimal solutions can be formulated in terms of finding a maximum-weight independent subset in a weighted matroid. That is, we are given a weighted matroid $M = (S, \mathcal{I})$, and we wish to find an independent set $A \in \mathcal{I}$ such that $w(A)$ is maximized. We call such a subset that is independent and has maximum possible weight an ***optimal*** subset of the matroid. Because the weight $w(x)$ of any element $x \in S$ is positive, an optimal subset is always a maximal independent subset—it always helps to make A as large as possible.

For example, in the ***minimum-spanning-tree problem***, we are given a connected undirected graph $G = (V, E)$ and a length function w such that $w(e)$ is the (positive) length of edge e. (We use the term "length" here to refer to the original edge weights for the graph, reserving the term "weight" to refer to the weights in the associated matroid.) We are asked to find a subset of the edges that connects all of the vertices together and has minimum total length. To view this as a problem of finding an optimal subset of a matroid, consider the weighted matroid M_G with weight function w', where $w'(e) = w_0 - w(e)$ and w_0 is larger than the maximum length of any edge. In this weighted matroid, all weights are positive and an optimal subset is a spanning tree of minimum total length in the original graph. More specifically, each maximal independent subset A corresponds to a spanning tree, and since

$$w'(A) = (|V| - 1)w_0 - w(A)$$

for any maximal independent subset A, an independent subset that maximizes the quantity $w'(A)$ must minimize $w(A)$. Thus, any algorithm that can find an optimal subset A in an arbitrary matroid can solve the minimum-spanning-tree problem.

Chapter 23 gives algorithms for the minimum-spanning-tree problem, but here we give a greedy algorithm that works for any weighted matroid. The algorithm takes as input a weighted matroid $M = (S, \mathcal{I})$ with an associated positive weight function w, and it returns an optimal subset A. In our pseudocode, we denote the components of M by $S[M]$ and $\mathcal{I}[M]$ and the weight function by w. The algorithm is greedy because it considers each element $x \in S$ in turn in order of monotonically decreasing weight and immediately adds it to the set A being accumulated if $A \cup \{x\}$ is independent.

Because B is optimal, A must also be optimal, and because $x \in A$, the lemma is proven. ■

We next show that if an element is not an option initially, then it cannot be an option later.

Lemma 16.8
Let $M = (S, \mathcal{I})$ be any matroid. If x is an element of S that is an extension of some independent subset A of S, then x is also an extension of $\emptyset$.

Proof Since x is an extension of A, we have that $A \cup \{x\}$ is independent. Since $\mathcal{I}$ is hereditary, $\{x\}$ must be independent. Thus, x is an extension of $\emptyset$. ■

Corollary 16.9
Let $M = (S, \mathcal{I})$ be any matroid. If x is an element of S such that x is not an extension of $\emptyset$, then x is not an extension of any independent subset A of S.

Proof This corollary is simply the contrapositive of Lemma 16.8. ■

Corollary 16.9 says that any element that cannot be used immediately can never be used. Therefore, GREEDY cannot make an error by passing over any initial elements in S that are not an extension of $\emptyset$, since they can never be used.

Lemma 16.10 (Matroids exhibit the optimal-substructure property)
Let x be the first element of S chosen by GREEDY for the weighted matroid $M = (S, \mathcal{I})$. The remaining problem of finding a maximum-weight independent subset containing x reduces to finding a maximum-weight independent subset of the weighted matroid $M' = (S', \mathcal{I}')$, where

$$\begin{aligned} S' &= \{y \in S : \{x, y\} \in \mathcal{I}\} , \\ \mathcal{I}' &= \{B \subseteq S - \{x\} : B \cup \{x\} \in \mathcal{I}\} , \text{ and} \end{aligned}$$

the weight function for M' is the weight function for M, restricted to S'. (We call M' the ***contraction*** of M by the element x.)

Proof If A is any maximum-weight independent subset of M containing x, then $A' = A - \{x\}$ is an independent subset of M'. Conversely, any independent subset A' of M' yields an independent subset $A = A' \cup \{x\}$ of M. Since we have in both cases that $w(A) = w(A') + w(x)$, a maximum-weight solution in M containing x yields a maximum-weight solution in M', and vice versa. ■

16.4-5

Show how to transform the weight function of a weighted matroid problem, where the desired optimal solution is a *minimum-weight* maximal independent subset, to make it an standard weighted-matroid problem. Argue carefully that your transformation is correct.

★ 16.5 A task-scheduling problem

An interesting problem that can be solved using matroids is the problem of optimally scheduling unit-time tasks on a single processor, where each task has a deadline, along with a penalty that must be paid if the deadline is missed. The problem looks complicated, but it can be solved in a surprisingly simple manner using a greedy algorithm.

A ***unit-time task*** is a job, such as a program to be run on a computer, that requires exactly one unit of time to complete. Given a finite set S of unit-time tasks, a ***schedule*** for S is a permutation of S specifying the order in which these tasks are to be performed. The first task in the schedule begins at time 0 and finishes at time 1, the second task begins at time 1 and finishes at time 2, and so on.

The problem of ***scheduling unit-time tasks with deadlines and penalties for a single processor*** has the following inputs:

- a set $S = \{a_1, a_2, \ldots, a_n\}$ of n unit-time tasks;
- a set of n integer ***deadlines*** $d_1, d_2, \ldots, d_n$, such that each d_i satisfies $1 \le d_i \le n$ and task a_i is supposed to finish by time d_i; and
- a set of n nonnegative weights or ***penalties*** $w_1, w_2, \ldots, w_n$, such that we incur a penalty of w_i if task a_i is not finished by time d_i and we incur no penalty if a task finishes by its deadline.

We are asked to find a schedule for S that minimizes the total penalty incurred for missed deadlines.

Consider a given schedule. We say that a task is ***late*** in this schedule if it finishes after its deadline. Otherwise, the task is ***early*** in the schedule. An arbitrary schedule can always be put into ***early-first form***, in which the early tasks precede the late tasks. To see this, note that if some early task a_i follows some late task a_j, then we can switch the positions of a_i and a_j, and a_i will still be early and a_j will still be late.

We similarly claim that an arbitrary schedule can always be put into ***canonical form***, in which the early tasks precede the late tasks and the early tasks are scheduled in order of monotonically increasing deadlines. To do so, we put the schedule into early-first form. Then, as long as there are two early tasks a_i and a_j finishing

				Task			
a_i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
w_i	70	60	50	40	30	20	10

Figure 16.7 An instance of the problem of scheduling unit-time tasks with deadlines and penalties for a single processor.

Proof Every subset of an independent set of tasks is certainly independent. To prove the exchange property, suppose that B and A are independent sets of tasks and that $|B| > |A|$. Let k be the largest t such that $N_t(B) \le N_t(A)$. (Such a value of t exists, since $N_0(A) = N_0(B) = 0$.) Since $N_n(B) = |B|$ and $N_n(A) = |A|$, but $|B| > |A|$, we must have that $k < n$ and that $N_j(B) > N_j(A)$ for all j in the range $k+1 \le j \le n$. Therefore, B contains more tasks with deadline $k+1$ than A does. Let a_i be a task in $B - A$ with deadline $k+1$. Let $A' = A \cup \{a_i\}$.

We now show that A' must be independent by using property 2 of Lemma 16.12. For $0 \le t \le k$, we have $N_t(A') = N_t(A) \le t$, since A is independent. For $k < t \le n$, we have $N_t(A') \le N_t(B) \le t$, since B is independent. Therefore, A' is independent, completing our proof that $(S, \mathcal{I})$ is a matroid. ■

By Theorem 16.11, we can use a greedy algorithm to find a maximum-weight independent set of tasks A. We can then create an optimal schedule having the tasks in A as its early tasks. This method is an efficient algorithm for scheduling unit-time tasks with deadlines and penalties for a single processor. The running time is $O(n^2)$ using GREEDY, since each of the $O(n)$ independence checks made by that algorithm takes time $O(n)$ (see Exercise 16.5-2). A faster implementation is given in Problem 16-4.

Figure 16.7 gives an example of a problem of scheduling unit-time tasks with deadlines and penalties for a single processor. In this example, the greedy algorithm selects tasks a_1, a_2, a_3, and a_4, then rejects a_5 and a_6, and finally accepts a_7. The final optimal schedule is

$$\langle a_2, a_4, a_1, a_3, a_7, a_5, a_6 \rangle ,$$

which has a total penalty incurred of $w_5 + w_6 = 50$.

Exercises

16.5-1

Solve the instance of the scheduling problem given in Figure 16.7, but with each penalty w_i replaced by $80 - w_i$.