Smallest Bookcase (Hard)

Problem category: Backtracking, Dynamic Programming
Expected difficulty: 3000

Solution

In this version of the problem, we need to partition the books into k shelves (where k is known from the solution of the easy version), such that L, the length of the longest shelf, is minimized.

If we fix a value of L, the question becomes: does there exist a partition of n books into k disjoint sets such that the following conditions hold?

- The size of each set does not exceed L;
- The books in a set are pairwise non-conflicting; and
- The union of all sets equals the original collection.

We can prove that this problem is NP-complete at best. For, if we set $L = n$ and $k = 1$, and assign infinite weight to the edges pertaining to conflicting pairs of books (in a complete graph whose nodes are the books), then the problem is effectively reduced to the decision version of the Traveling Salesman Problem, which is known to be NP-complete.

We know that a dynamic programming solution to the TSP exists, namely, the Held-Karp algorithm, which runs in $O(n^2 2^n)$ time. However, we'd have to adapt that solution to our general case, which is harder. In our attempts, we devised a $O(4^n)$ solution based on it, but it isn't fast enough for $n > 13$.

Instead, let's try a more direct approach. We start with a naive backtracking solution and incrementally improve it until it's good enough for all $n \leq 18$.

First, note that the number of possible partitions of n elements into k parts is given by the Stirling number of the second kind:

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n$$

The worst case for $n = 18$ is when $k = 7$, which results in about $1.97 \cdot 10^{11}$ partitions. However, this can be improved if we remove some books from the whole set. But which ones?

Well, we know that conflicting books must be kept in separate shelves, and that the largest set of conflicting books has size k. So, if we fix k conflicting books as an initial configuration, there remains $n - k$ books to partition into k parts. Thus, we reduce the number of possibilities to

$$\left\{ {n-k \atop k} \right\}$$

Now, the worst case would be when $k = 4$, which results in 10391745 partitions. This still might not be good enough, because we need to perform a lot of checks before arriving at a valid partition. So we need a way to reduce the search space even further. Can we do it?

Indeed we can. Note that we are trying to minimize L. It turns out that if at some point during the search we encounter a shelf of length equal to our best answer so far, we can stop right there and backtrack, since we will no longer be able to improve the answer. This avoids an entire branch of the search tree and significantly speeds up the computation.

Thus, although we cannot guarantee an analytical upper bound on the running time, a carefully implemented backtracking algorithm would yield good enough performance for our purposes.