LECTURE 14 – TACKLING NP-COMPLETE PROBLEMS

Overview

- TACKLING NP-complete PROBLEMS
- Branch and Bound for Knapsack
- Bin Packing Approximation Algorithms

TACKLING NP-complete PROBLEMS

There are two approaches:

1. Search techniques that although of exponential time complexity, are an improvement over exhaustive search. These include techniques that construct partial solutions such as branch and bound and techniques that reduce the worst time complexity by making clever choices while searching the tree such as specific algorithms for some NP-complete problems.

2. Search techniques that do not aim to find the optimal solutions but a good enough one. Examples of these techniques are local search heuristics.

A combinatorial optimisation problem (maximisation or minimisation) $\prod$ consists of three parts:

1. A set of instances D

2. For each instance I a finite set $S_D(I)$ of candidate solutions to I

3. An evaluation function F that assigns to each instance I and each candidate solution $s \in S_D(I)$ a positive rational number F(I,s) called the solution value of s.

If $\prod$ is a maximisation problem then:

An optimal solution s* is one that for all $s \in S_D(I)$, $F(I,s^*) \geq F(I,s)$ then OPT(I) is used to denote the value for an optimal solution to the instance I.

An algorithm A is called an approximation algorithm if it finds a solution $s \in S_D(I)$ with value A(I). The algorithm is called an optimisation algorithm if A(I)=OPT(I).

If $\prod$ is NP-complete, no polynomial time algorithm can be found unless **P**=**NP**.

The option then is to find an approximation algorithm A that runs in 'reasonable time' and for each instance I of $\prod$, A(I) is 'close' to OPT(I).

BRANCH AND BOUND FOR KNAPSACK

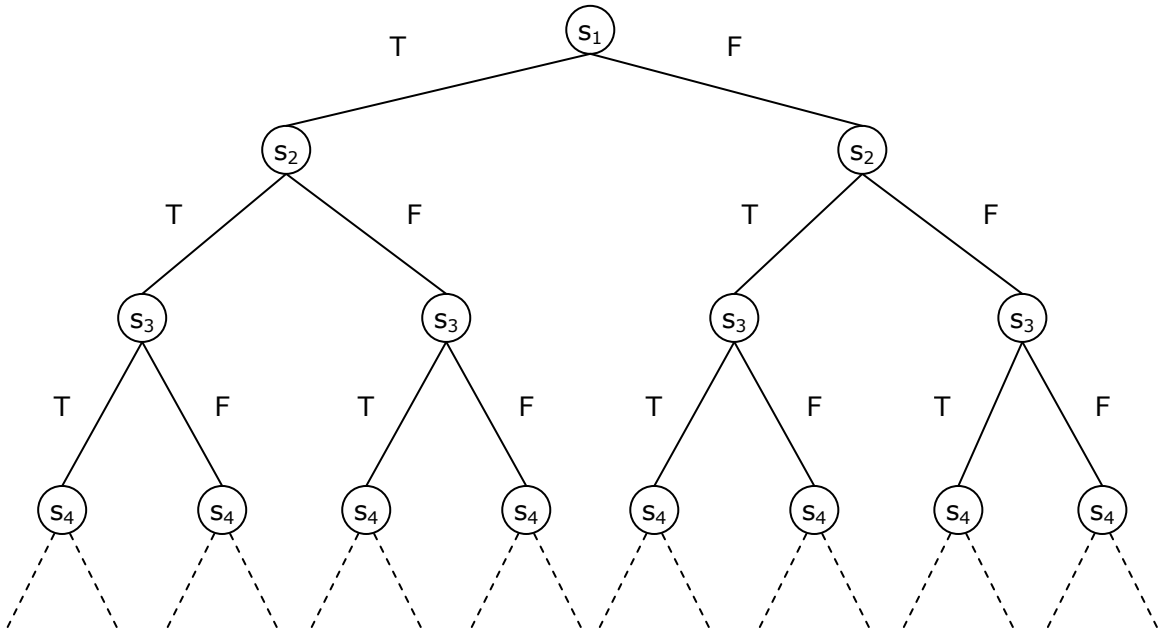
The Branch and Bound algorithm is basically a tree search approach based on enumeration in which non-promising parts of the search space are pruned. This algorithms uses three procedures:

Branching – the feasible search space represented as a tree structure is divided into sub-regions and hence the whole search space can be covered recursively

Bounding – finding upper and lower bounds for the optimal solution within a given feasible sub-region (part of the search tree)

Pruning – for a maximisation problem, if the upper bound for a sub-region A of the search tree is greater than the lower bound for any other previously examined sub-region B, then sub-regions A may be safely discarded (pruned) from the search tree.

The total enumeration of solutions for a Knapsack optimisation in which the goal is to maximise the profit by selecting the items to put into the knapsack, can be represented in a tree structure as follows where each node in the tree represents the decision whether to include an item in the knapsack.



An incumbent solution can be computed which is the reference solution and can be used as the lower bound.

Then at any point of the search, i.e. in every node, a decision can be made as to whether search or prune the remainder of the tree depending of the upper bound calculated in the node.

The performance of a Branch and Bound algorithm depend of how the branching is made and how tight are the computed upper bounds (for maximisation).

BIN PACKING APPROXIMATION ALGORITHMS

For the BIN PACKING problem with a set of items $U=\{u_i,u_2,\ldots,u_n\}$ each with size given by a rational number s_i and a set of bins each with capacity equal to 1. The optimisation problem is to find a partition of U into disjoint subsets $U_1, U_2,\ldots,U_k$ such that the sum of sizes of the items in each U_i is no more than 1 and such that k is as small as possible. That is, the goal is to pack all the items in as few bins as possible.

The 'First Fit' (FF) algorithm is an approximation algorithm for BIN PACKING:

- given the sequence of unit-capacity bins $B_1,B_2,B_3\ldots$
- place the items into the bins, one at a time, item u_i is placed in the lowest-indexed bin with enough remaining capacity for s_i which is the item size

It has been proven that the worst performance: $FF(I) \leq 1.7\ OPT(I) + 2$

The 'Best Fit' (BF) algorithm is another approximation algorithm for BIN PACKING:

- given the sequence of unit-capacity bins $B_1,B_2,B_3\ldots$
- place the items into the bins, one at a time, item u_i is placed in bin with enough remaining capacity closest to but not exceeding $1 - s_1$

It has been proven that the worst performance: $BF(I) \leq 1.7\ OPT(I) + 2$

The performance of the FF and BF approximation algorithms for BIN PACKING depends very much on the ordering of the items, in particular whether larger or smaller items appear before in the sequence.

The 'First Fit Decreasing' (FFD) algorithm is another approximation algorithm for BIN PACKING:

- given the sequence of unit-capacity bins $B_1,B_2,B_3\ldots$
- order the items in non-increasing order of their size
- place the items into the bins, one at a time, item u_i is placed in the lowest-indexed bin with enough remaining capacity for s_i which is the item size (as in FF)

It has been proven that the worst performance: $FFD(I) \leq 1.22\ OPT(I) + 4$

Exercise 14.1. Given the sets of items:

$U_1=\{0.1,0.15,0.5,0.2,0.6,0.4,0.3,0.25,0.5,0.35,0.10,0.4\}$

Illustrate the working of the FF, BF and FFD algorithms for BIN PACKING.

Reading: (Garey&Johnson, Chapter 6)