Integer Programming

Guillaume Dalle (ENPC, CERMICS)

REOP - Class 6 (10/11/2021)

Contents

1 Homework

Exercise: Dynamic programming for the knapsack problem

The knapsack problem consists in filling a bag by choosing from a set of items $i \in \{1, ..., n\}$ with weights w_i and prices p_i. Its decision version asks whether there is a subset with total weight $\leq W_{\max}$ but total price $\geq P_{\min}$. Propose a dynamic programming algorithm to compute the value function

$$P(n, W) = \text{maximum price for a bag of weight} \leq W \text{ with the } n \text{ first items}$$

Is it enough to prove that the knapsack problem is in **P**?

Exercise (Meunier - PL6): Computing sums of order statistics

For any vector $x \in \mathbb{R}^n$, we denote by $x^{[i]}$ the i-th largest element of the set $\{x_1, ..., x_n\}$. For instance, $x^{[1]}$ is the largest component of x, and $x^{[n]}$ is the smallest.

1. Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We fix an integer $k \in [n]$. Show that the problem

$$\min \sum_{i=1}^{k} x^{[i]} \quad \text{s.t.} \quad Ax = b, x \in \mathbb{R}^n_+$$

   can be formulated as a Linear Program. Hint: introduce one constraint per subset $S \subset [n]$ of size k.
2. (Harder) Show that there is a formulation with at most $m + n$ constraints.

2 Modeling with linear functions and integrality constraints

2.1 Mixed Integer Linear Programs and their relaxations

A MILP is an LP where some variables are forced to take integer values:

$$\min_{x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}} c^\top x \quad \text{s.t.} \quad Ax \leq b \tag{P}$$

The continuous relaxation of a MILP is the LP obtained by removing integrality constraints:

$$\min_{x \in \mathbb{R}^n} c^\top x \quad \text{s.t.} \quad Ax \leq b \tag{R}$$

Examples:

- $x \in \mathbb{N}$ becomes $x \geq 0$
- $x \in \{0, 1\}$ becomes $0 \leq x \leq 1$

If x is a solution of (P), then it is a solution of (R). As a consequence, $\text{sol}(P) \subset \text{sol}(R)$ and $\text{val}(P) \geq \text{val}(R)$ (because taking the infimum over a smaller set leads to a larger value).

Complexity:

- LPs are easy to solve, even in very high dimension
- MILPs are hard to solve, they require solving a large number of associated LPs

Exercise (Meunier - PL11): Basic principle of column generation

Let (P) be the linear program

$$\min c^\top x \quad \text{s.t.} \quad Ax = b, x \in \mathbb{R}^n_+$$

We suppose that n is very large. Let $I \subset [n]$, we consider the program (P^I):

$$\min c_I^\top x' \quad \text{s.t.} \quad Ax' = b, x' \in \mathbb{R}^I_+$$

We denote by (D) and (D^I) the respective dual problems of (P) and (P_I). Let $\tilde{y}$ be an optimal solution of (D^I). Show that if $\tilde{y}$ is also a feasible solution to (D), then the optimal value of (P^I) is the same as the optimal value of (P).

Why can't we just solve the relaxation and round the solution to the nearest integer?

- Rounding is hard in high dimension: 2^d possibilities
- Sometimes none of the closest integer solutions are feasible
- There can be an arbitrary gap between the value of (R) and the value of (P)

Exercise (Meunier, PL12): 2-approximation of VERTEX COVER with an LP

1. Find a MILP formulation for the minimum vertex cover problem
2. Propose a way to build a vertex cover from a solution to the linear relaxation of the previous MILP.
3. Prove that the solution you obtain at at most twice as large as the minimum vertex cover.

2.2 Concrete examples

See (Williams 2013) for very useful tricks in model building.

2.2.1 Dispatching patients during a pandemic

Consider a set of H hospitals with reanimation capacity r_h and initial patient load p_h. An excess patient in h costs e_h, and a patient transfer from h_1 to h_2 costs $t_{h_1 h_2}$. Formulate a MILP to optimize patient dispatch.

2.2.2 The power of logical variables

Exercise 10.4

Exercise 10.5

3 Integral polyhedra

3.1 When MILPs become easy to solve

A polyhedron P is integral if $P = \text{conv}(P \cap \mathbb{Z}^n)$. In particular, all its vertices have integer coordinates, which means that

- any LP with P as feasible set has one integral optimal solution
- the simplex algorithm finds such a solution

This implies that solving a MILP is just as easy as solving its continuous relaxation.

3.2 Total Unimodularity

A matrix A is Totally Unimodular (TU) if the determinant of all its square submatrices (obtained by removing rows and/or columns) is in $\{-1, 0, 1\}$. In particular, all of its coefficients must be in $\{-1, 0, 1\}$ too.

Theorem 9.8 - variant (Importance of TU matrices): If A is TU and $b \in \mathbb{Z}^m$, then $P = \{Ax = b, x \geq 0\}$ is integral.

Proof: For any vertex x of P, there is a basis B such that x is the basic feasible solution associated with B. This implies that $x_N = 0$ and $x_B = A_B^{-1}b \geq 0$. Since A is TU and A_B is invertible, the determinant of A_B is in $\{-1, 1\}$. By the comatrix formula, $A_B^{-1} = \frac{1}{\det A_B}\text{com}(A)^\top$, which means A_B^{-1} has integer coefficients. Since b does too, we conclude that $x \in \mathbb{Z}^n$.

Proposition 9.10 (Poincaré criterion for TU): Let $A \in \{-1, 0, 1\}^{m \times n}$. If A contains at most one 1 and one -1 per column, then A is TU.

Proof: By recursion.

Corollary 9.11 (Integrality of the flow polyhedron): The incidence matrix of a direted graph is TU.

How to build new TU matrices? If A is TU, then the following matrices are too: $A^\top$, $-A$, $(A, \pm A)$, (A, e_j), any matrix obtained multiplying a row or column of A by -1.

3.3 Examples

Exercise (Meunier, PLNE1)

Exercise (Meunier, PLNE6)

4 Solution algorithms

Naive approach:

1. Enumerate all possible assignments of integer variables
2. For each of them, solve the resulting LP
3. Pick the best solution

Unfortunately, this is very inefficient.

4.1 Branch & Bound

Branch & Bound speeds up enumeration by discarding useless cases. It is still exponentially slow in the worst case, but it works very well in practice.

B&B starts by solving the relaxation (R). If the solution x^* has only integer coordinates, bingo, we stop there. Otherwise, we select a fractional coordinate x_j^* and divide the search space by adding one of two constraints: $x_j \leq \lfloor x_j^* \rfloor$ or $x_j \geq \lceil x_j^* \rceil$. Then we iteratively apply the same procedure to these two smaller MILPs, thus building a binary tree where each node corresponds to a subregion of the search space.

To avoid enumerating all possible assignments, we have pruning techniques:

- Prune by infeasibility: If the relaxation at a node has no continuous solution, then we don't need to keep looking, because there will be no integer solution either in this region of the space.
- Prune by optimality: If the relaxation at a node returns an integer solution, then we don't need to keep looking because we already know the best candidate in this region of the space.
- Prune by bound (the most important): If the relaxation at a node has a value higher than the best solution found sofar (in the whole tree), then we don't need to keep looking because no matter what we find in this region of the space, it will be worse than the best candidate we already have.

Exercise 10.3 (approximate B&B)

Tuning Branch & Bound is often necessary, and commercial solvers include many complicated heuristics meant to speed up the search. Among the most important design choices, we can list:

- Choice of the variable to branch on
- Choice of the next node to explore

Practical illustration in Julia

Open this webpage and complete the code cells to manually apply B&B.

Note that Branch & Bound is a general principle which applies beyond Integer Programming (see course notes for a more abstract formulation of the algorithm).

4.2 Cutting planes

TBC

References

Williams, H. Paul. 2013. *Model Building in Mathematical Programming.* John Wiley & Sons. http://books.google.com?id=YJRh0tOes7UC.