(DRAFT) Implicit enumeration with dual bounds obtained from approximation algorithms

Nelson Frew

Introduction

The field of discrete optimisation (CITE) studies problems that model discrete decision making. Mixed-Integer Programming refers to a set of modelling and solving techniques for such discrete problems (CITE) by formulating them as Mixed-Integer Programs (MIPs). For many problems, this approach is the most efficient known method to find a solution. The standard MIP formulation, for maximisation, is as follows:

$$\begin{array}{ll} \text{maximise} & cx + hy \\ \text{subject to} & Ax + Gy \leq b \\ & x \geq 0 \text{ integral} \\ & y \geq 0 \end{array}$$

Generally, objective functions are constrained by linear inequalities, and variables can be specified to be integer or continuous.

Optimisation problems with linear constraints and continuous variables are known as Linear Programs (LPs), which are both solvable in polynomial time and very efficiently in practice (CITE). In this report, we consider maximisation problems whose constraints are all linear. MIPs, on the other hand, are characterised by having one more more variables constrained to be integer, generally resulting in an NP-Hard problem (CITE). By removing the integer constraints on variables in a MIP, we obtain its associated LP relaxation.

The state-of-the-art method for solving MIPs leverages this relationship to systematically search for what are known as primal solutions that are feasible, using bounds to implicitly enumerate through these.

Because the feasible region of continuous solutions naturally subsumes the set of feasible solutions for integers, we know, for optimal value for the LP relaxation z_{LP} and true optimal value z, that $z \leq z_{LP}$.

$$z \leq z_{LP} \tag{1}$$

Since z_{LP} is at least as large as our optimal value, we are provided with an upper bound in our case, which we refer to as the *dual bound*. Given our solved relaxation, we form a subproblem by adding constraints to our problem which constrain certain variables to be integer, and solve this. The process of choosing a given variable to constrain is known as *branching*. By solving LP relaxation again for this constrained subproblem, we can see if an optimal integer solution can exist in with these constraints: if our dual bound is lower than our best known value so far, we can prune this subproblem and switch to another constrained subproblem. Otherwise, if this subproblem has a dual bound above our best known value, we continue constraining and solving the relaxation until a new feasible solution is found, or a further constrained subproblem of this is pruned.

Should I note that x and y are vectors of assigned values within the objective?

This constitutes the core of the Branch and Bound approach, which we formally describe below: let (x^i, y^i) be the optimal solution to linear program LP_i, (x^*, y^*) denote an optimal solution to the MIP; $\underline{z}$ denote the

lower bound on the optimal value and z^* the optimal solution of the MIP, and let $\mathcal{L}$ denote the list of nodes of the Branch and Bound tree yet to be solved (i.e. not pruned nor branched on).

Algorithm 1: Branch and Bound algorithm for Mixed-Integer Programming

Data: A Mixed-Integer Program.
Result: The optimal solution value, and the associated solution

```
Set L := N_0, set z := −∞, set (x*, y*) := ∅;
while L is not empty do
    Select a node N_i from L, by some node selection scheme, deleting it from L;
    Solve LP_i to get solution value z_i and (x^i, y^i) if it exists;
    if LP_i is infeasible, i.e. z_i = −∞ then
        Go to step 2;
    end
    else
        Let (x^i, y^i) be an optimal solution of LP_i and z_i be its objective value
    end
    if z_i ≤ z then
        Go to step 2;
    else
        if (x^i, y^i) is feasible to the MIP then
            Set z := z_i;
            Set (x*, y*) := (x^i, y^i);
            Go to step 2;
        end
    end
    else
        Branching: From LP_i construct 2 linear programs LP_i1, LP_ik with smaller feasible regions whose
          union does not contain (x^i, y^i), but contains all the solutions of LP_i with x ∈ ℤ, by choosing a
          variable to constrain.;
        Add the corresponding new nodes N_i1, N_ik to L and go to step 2.
    end
    Return z;
end
```

While finding optimal solutions with Branch and Bound is in worst case exponential-time, for some problems there exist Approximation Algorithms (AAs) which can provide feasible solutions in as fast as polynomial-time, by sacrificing guaranteed optimality. In such cases, AAs provide a guarantee on the *quality* of the solution, within some factor.

An α-approximation is an AA that guarantees that the solution value will always be within a constant factor $\alpha < 1$ of the optimal solution; i.e. for an approximate solution value z_A, it must be that $\alpha z \leq z_A \leq z$. As a result, for a maximisation problem, we can analytically derive an upper bound on the optimal value from the lower bound guarantee on the optimal solution. In this way, we can use AAs in lieu of an LP relaxation to obtain valid dual bounds on our optimal solution. Further, if we use information from a found approximate solution, we can derive improved dual bounds *a posteriori* on our optimal solution. Because we are leveraging guarantees on the optimal value, the dual bounds we obtain are also guaranteed to be within a level of accuracy. This is distinct from LP relaxations, which can provide arbitrarily poor dual bounds.

When conducting a Branch and Bound with AAs, an important observation is that the general strategies which have contributed to the success of the standard Branch and Bound with LPs must be addressed. In particular, methods of branching and node selection do not have a clear translation with respect to AAs. In this project, we investigate methods of effectively using dual bounds provided by AAs for the 0,1 Knapsack. To do this, we investigate known approximation schemes and methods to construct high quality dual bounds from them, as well as devising branching strategies within its Branch and Bound.

In the following sections, we give a brief literature review of the research and improvements to the Branch and

Bound approach, as well as an overview of details and approaches with AAs that are central to this report.

(**Put warm starting in future work**)

Branch and Bound improvements and strategies

While our previous description of the Branch and Bound approach captured the core components, there still remain many aspects which are uncertain: which variables to constrain for branching, alternative methods to relax a solution, and node selection schemes. These problems are in fact non-trivial and represent active research areas within Mixed-Integer Programming. In light of this, we provide an overview of known strategies and the body of research that comprises the current understanding in the field.

Branching Strategies

Deriving useful subproblems is central to the Branch and Bound approach, as it provides us with the partial enumeration required to establish bounds on an optimal solution value. A basic strategy is to branch on the variable with the largest fractional component, however [1] has shown this to be as bad as random selection. It is possible to make inferences based on the properties of each variable's fractional component, as shown by [5] by the proposal of "use penalties". This was used to inform the branching process where "up penalties" and "down penalties" were calculated for each variable based on their respective fractional components, after which the variable with the highest penalty in any direction was chosen. However, [3] showed with computational experiments that such approaches may have limited use, while approaches involving issuing "pseudocosts" as proposed by [2] may have advantages.

In an attempt to provide a categorisation for branching methods, [4] proposed two groupings: *binary/non-binary* strategies, and *wide* strategies. Both problem types and tree depth provided the ideas central to these classifications.

A detailed survey of the literature and the current open questions in research are provided by [4], where the reader is directed for further reading.

Bounding optimal solutions

Extensions, improvements, and related techniques

Stuff about approximations

All the big Knapsack approximations.

Probably introduce both KP and approximations notation here. We'll probably always need:

- Error parameter ϵ
- Profits

Sahni (1975)

Ibarra and Kim (1975) (Williamson Shmoys + Vazirani)

Lawler (1979)

Magazine and Oguz (1981)

Kellerer and Pferschy (1999)

Kellerer and Pferschy (2001)

Anything else?

Design and Analysis

Implementation and results

What is a logical way to lay all this out?

References

[1] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[2] Michel Bénichou, Jean-Michel Gauthier, Paul Girodet, Gerard Hentges, Gerard Ribière, and O Vincent. Experiments in mixed-integer linear programming. *Mathematical Programming*, 1(1):76–94, 1971.

[3] Gautam Mitra. Investigation of some branch and bound strategies for the solution of mixed integer linear programs. *Mathematical Programming*, 4(1):155–170, 1973.

[4] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[5] JOHN A Tomlin. Branch and bound methods for integer and non-convex programming. *Integer and nonlinear programming*, 1:437–450, 1970.